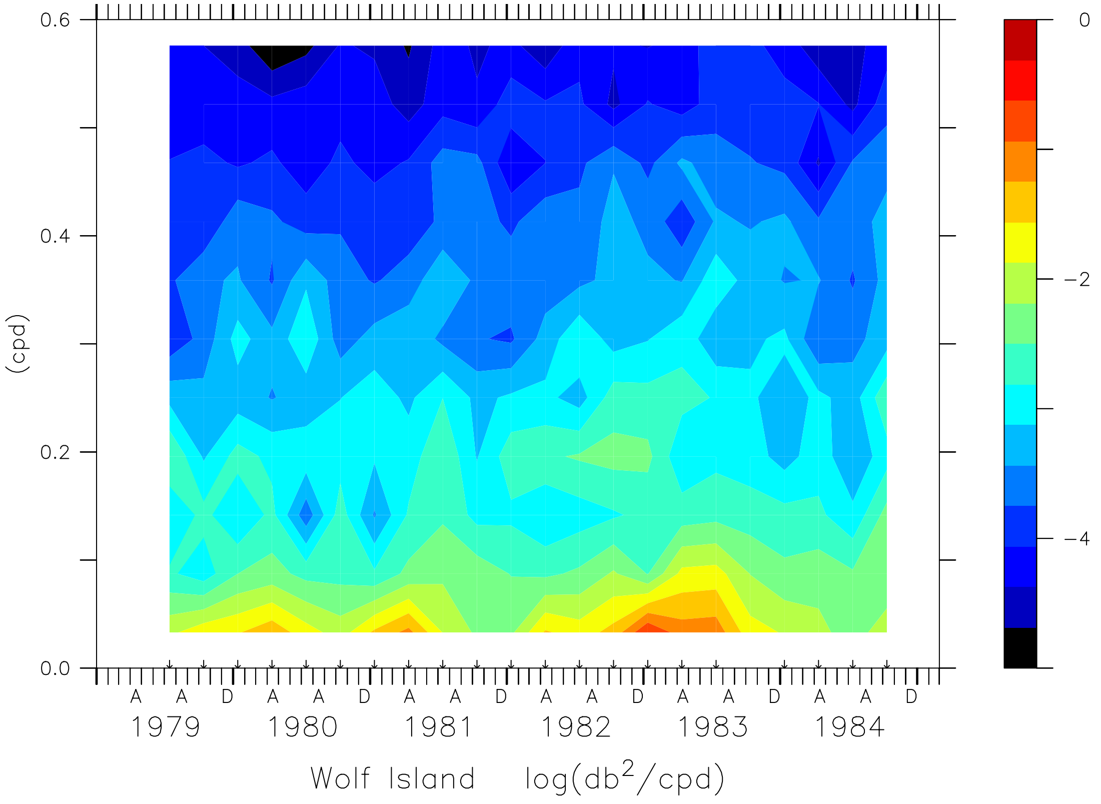
Plot Plus
User’s Guide

Version 1.3.3
January 2000



Plot Plus Graphics

2138 N 186th St
Shoreline, Washington 98133
Phone/Fax: (206) 366-0624
e-mail: plot_plus@halcyon.com
http://www.halcyon.com/www2/dwd/

Plot Plus Licensing Information

November 7, 1996

Introduction

Plot Plus Graphics has made its software for the 1.X release of Plot Plus available free of charge upon application. There may be software that uses Plot Plus which is developed outside of Plot Plus Graphics. The source code for such material may or may not be available, at the copyright holder's option.

Use of the Plot Plus software, or derivative works based thereon, within a commercial product, for a commercial purpose, or a within a product distributed for any fee or handling charge REQUIRES LICENSING,which may include a fee payment. Plot Plus Graphics will negotiate commercial-use licenses for Plot Plus upon request. These requests can be directed to plot_plus@halcyon.com.

Applying

Individuals may apply in their own name. A separate license is required for each platform on which Plot Plus will run, and a separate license is required to bundle Plot Plus with another application, either by directly calling Plot Plus as a subroutine or used as a non-standalone application.

Individuals or organizations that hold licenses for older versions of Plot Plus must get a new license in order to download or use a new version of Plot Plus.

PLOT PLUS GRAPHICS PLOT PLUS SOFTWARE LICENSE AGREEMENT

Upon execution of this Agreement by the party identified below ("Licensee"), Plot Plus Graphics provides the Plot Plus (TM) software in Binary and/or Source Code form ("Software") to Licensee, subject to the following terms and conditions. For purposes of this Agreement, Binary is the compiled code which is ready to run on Licensee's computer. Source code consists of a set of files which contain the actual program commands that are compiled to form the Binary. Plot Plus is considered to be "bundled" with another application when Plot Plus is either used as a subroutine or in tight coupling with another application via shell scripts, RPC calls, batch files, etc.

1. The Software is intellectual property owned by Plot Plus Graphics, and all right, title and interest, including copyright, remain with Plot Plus Graphics. Plot Plus Graphics grants, and Licensee hereby accepts, a restricted, non-exclusive, non-transferable license to use the Software for academic, research and internal business purposes only, without a fee. Licensee agrees to reproduce the copyright notice and other proprietary markings on all copies of the Software. Licensee has no right to transfer or sublicense the Software to any unauthorized person or entity. However, Licensee does have the right to make complimentary works that interoperate with Plot Plus, to freely distribute such complimentary works, and to direct others to the ftp server to obtain copies of Plot Plus itself.

2. Licensee may, at its own expense, modify the Software to make derivative works, as necessary for its own academic, research, and internal business purposes. However, Licensee's distribution of any derivative work is also subject to the same restrictions on distribution and use limitations that are specified herein for Plot Plus Graphics's Software, and may not be distributed to third parties without an appropriate license from Plot Plus Graphics. Any derivative work should be clearly marked to notify users that it is a modified version and not the original Plot Plus distrib- uted by Plot Plus Graphics.

3. PLOT PLUS GRAPHICS MAKES NO REPRESENTATIONS ABOUT THE SUITABILITY OF THIS SOFTWARE FOR ANY PURPOSE. IT IS PROVIDED "AS IS" WITHOUT ANY EXPRESS OR IMPLIED WARRANTY. PLOT PLUS GRAPHICS SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED BY THE USERS OF THIS SOFTWARE.

Licensee understands the Software is a research tool for which no warranties as to capabilities or accuracy are made, and Licensee accepts the Software "as is". Licensee assumes the entire risk as to the results and performance of the Software and/or associated materials. Licensee agrees that Plot Plus Graphics shall not be held liable for any direct, indirect, consequential, or incidental damages with respect to any claim by Licensee or any third party on account of or arising from this Agreement or use of the Software and/or associated materials.

4. Licensee understands the Software is proprietary to Plot Plus Graphics. Licensee agrees to take all reasonable steps to insure that the Source Code is protected and secured from unauthorized disclosure, use, or release and will treat it with at least the same level of care as Licensee would use to protect and secure its own proprietary computer programs and/or information, but using no less than a reasonable standard of care. Licensee agrees to disclose the Software to any other person or entity with a need to know, on the terms and conditions as specified in this agreement. If Licensee becomes aware of any unauthorized licensing, copying or use of the Software, Licensee shall promptly notify the Plot Plus Graphics in writing. Licensee expressly agrees to use the Software only in the manner and for the specific uses authorized in this Agreement.

5. By using or copying this Software, Licensee agrees to abide by the copyright law and all other applicable laws of the U.S. including, but not limited to, export control laws and the terms of this license. Plot Plus Graphics shall have the right to terminate this license immediately by written notice upon Licensee's breach of, or non-compliance with, any of its terms. Licensee may be held legally responsible for any copyright infringement that is caused or encouraged by its failure to abide by the terms of this license. Upon termination, Licensee agrees to destroy all copies of the Software in its possession and to verify such destruction in writing.

6. In all uses of the Software, where appropriate, Licensee will credit the origins of the Software to Plot Plus Graphics for its role in the development of the Software.

(i) If Plot Plus is bundled with another program, Licensee will credit the origins of the Software to Plot Plus Graphics at every instance where the bundling program is identified.

(ii) Idenification of Plot Plus can not be removed or disabled from the startup banner of Plot Plus when used as a standalone application or the Plot Plus source code.

7. Should Licensee wish to make commercial use of the Software, Licensee will contact Plot Plus Graphics (plot_plus@halcyon.com) to negotiate an appropriate license for such use. Commercial use includes (1) integration of all or part of the Software into a product for sale or license by or on behalf of Licensee to third parties, or (2) distribution of the Software to third parties that need it to utilize a commercial product sold or licensed by or on behalf of Licensee. For example:

(i) Any automatic assistance for downloading and/or installing the Software requires a commercial license. If the Licensee is providing a service that assists a user in transferring the Software to their machine or installs the Software on a user's machine, then a commercial license is required.

(ii) Distribution of the Software through such formats as diskette and CD-ROM requires a commercial license.

8. Plot Plus Graphics is strongly interested in receiving information related to the performance of the Software in user's computing environment, and encourages contributions from users of the code that might, at Plot Plus Graphics's sole discretion, be used or incorporated to make the Software a more stable, flexible, and/or useful product. Licensees that wish to contribute their code must sign an "Agreement Regarding Contributory Code for Plot Plus Software" before Plot Plus Graphics can accept it (contact plot_plus@halcyon.com for a copy).

UNDERSTOOD AND AGREED: LICENSEE

Personal Information:

Please fill out the following fields. Some of these are mandatory to obtain a valid license: Name, organization, physical address, telephone number, machine name.

1. First Name: ______________________
2. Last Name (Family Name): _______________________
3. Telephone Number: ___________________
4. Title: ________________________
5. Organization - including department or workgroup:______________________________
____
6. Email:__________________________________
7. University/Company/Agency Address: ____
____
____
____
____
8. Host name:_______________________________
9. Bundling software name (if applicable):____

Other Information:
1. Intent for use of the software (Please be specific and detailed. Give URLs where applicable.):
____
____
____
____
____
____
____
____
____
____
2. Are you willing to contribute your modifications/enhancements of the code, or associated applications for inclusion in future Plot Plus releases? Check one:____Yes____No.

Contact Information:

The best contact path for licensing issues is by e-mail to “plot_plus@halcyon.com”

Send mail to:
Plot Plus Graphics
2138 N 186th St
Seattle, WA 98133 USA
FAX: (206) 366-0624

Table of Contents

Preface

Y2K Compliance

The present version of *PPLUS* has had all dependencies on 2 digit years removed, while maintaining compatibility with older pre-Y2K scripts. When specifying dates, if a string of the form “Wyymmddhhmm” is used, 1900 will be added to *yy*. If a string of the form “Wyyyymmddhhmm” is used, *yyyy* will be used unmodified. The time axes have been updated to display the full four digit year and all internal time calculation are done with four digit years.

Changes since *PPLUS* version 1.2x

PPLUS version 1.3.2 has many bug fixes as well as several new features.

- **pltnmext** command has been added. Used in conjunction with the **pltnme** command it is now possible to produce file names with a standard file name extension.
- **flush** command has been added. **flush** causes any partially create meta file blocks to be written to the plot file.
- uses 4-digit years for all “internal” dates
- Can use either WYYMMDDHHMM or WYYYYMMDDHHMM for date input on command line (*PPLUS* assumes any 2-digit year given is 19XX!)
    - W9101011200 will be interpreted as Jan 1, 1991 12:00
    - W0002271300 will be interpreted as Feb 27, 1900 13:00
    - W200112271434 will be interpreted as Dec 27, 2001 14:34
    - W199901050000 will be interpreted as Jan 5, 1999 00:00
- *PPLUS* user function supports 4-digit years (remember to use i4 in the format statement!)
- will continue to read classic epic files (again *PPLUS* assumes that the 2-digit years mean 19xx!!!
- time axes (both x and y) have been expanded to handle the 4-digit years (you can specify dates to the end of the decade now).
- e function has been extended to handle 4-digit years in the time specification. e.g. sl=[t=200102012230]

Differences between PLOT5 and *PPLUS*

PPLUS is a greatly enhanced replacement to PLOT5. Most PLOT5 syntax and commands are identical to *PPLUS* usage. However, there are the following differences and incompatibilities.

- **rdcom** command has been replaced by the @ command.
- The **lev** command replaces the **level** and **cline** commands.
- In format statements and labels single quotes (‘) must be replaced by two single quotes (‘ ’). The same applies to double quotes (“). See the chapter on labels.

- The **limits** command is enhanced.
- **if** / **else** / **endif** and **while** / **endw** logic are available in command files. The **inc** and **dec** commands are available to increment and decrement symbols.
- The **txlint**, **txlabp**, **txlsze**, **txnmtc** and **txtype** commands should be used instead of using the corresponding arguments in the **taxis** command.
- The **time** command should be used instead of the **tmin**, **tmax** and **tstart** commands.
- The **velvct** command is replaced by the **vector**, **vecset** and **veckey** commands.

NOTE: The following commands *are not* supported in this and future versions of *PPLUS*:

- **tmin**, **tmax** and **tstart**
- **level** and **cline**
- **rwdseq**, **readseq** and **skpseq**
- **taxis** does not support the obsolete arguments.
- **velvct**

Conventions in Text

The following conventions are used in this manual to display information.

- The `plain typewriter font` is used to show text that is entered into *PPLUS* verbatim.
- The **bold helvetica font** is used to denote PPLUS commands or text that should be entered exactly as shown.
- The *italic helvetica font* or *`italic courier font`* indicates variables or parameters that you should replace with an appropriate word, string, or number.
- The *italic font* is used for emphasis.
- Optional command elements are enclosed with square brackets [ ].
- The notation { **on** | **off** } indicates that either **on** or **off** should be entered as a command argument.

1 Introduction

Plot Plus (*PPLUS*) is an interactive, command-driven general-purpose program for plotting user supplied data. *PPLUS* recognizes data in standard fortran formatted, unformatted and free format files as well as some specialized formats (see the chapter on Data Formats). Data can also be entered from the keyboard.

The major use of *PPLUS* is the plotting of contour data and X-Y pairs. A very small number of commands are required to generate a plot, making use of the many defaults available. However, it is also possible to control almost every aspect of the plot and to generate a final product which looks as though it were professionally drafted. Over thirty character sets are available, including special Greek and Math symbols. It is possible to make a composite of several plots of different kinds (or the same kind) on a single page and to add text information anywhere on the plot.

PPLUS commands can be entered interactively from the keyboard or from a command file much like a VAX/VMS command file. *PPLUS* command files support parameter passing, symbol substitution, and logic structures such as **while** loops and block **if** statements. The *PPLUS* command files are simple ASCII disk files which are easily edited with a text editor.

On-line help is available with the VAX/VMS command `HELP PPLUS`. (First, *PPLUS* definitions must have been established as indicated in “Getting Started” on page 3.)

2 Getting Started

UNIX

Environment variables

PPLUS requires an environment variable to be defined to correctly operate a wide range of interactive graphics devices. The following should be included in your `.login` file.

```
setenv PPL_HOME pplus_home_directory
```

where "*`pplus_home_directory`*" is the root directory where *PPLUS* is installed. *PPLUS* uses the following directory structure.

pplus_home_directory

/bin	executables for pplus and filters
/lib/pplusfonts	plot plus fonts directory
/scripts	system pplus scripts, directory is appended to PPL_COMMAND_PATH
/pal	colormap (palette) directory
/lib/epic.key	epic key file default location.

The command "**loadcmap**" will look in pplus_home_directory/pal directory for color maps.

```
setenv PPL_STARTUP path_name
```

where "*`path_name`*" is a file containing *PPLUS* commands used to initialize your *PPLUS* environment. The command file is executed each time you begin *PPLUS*.

```
setenv PPL_COMMAND_PATH pplus_script_path
```

where "*pplus_script_path*" is a list of directories separated by colons where *PPLUS* scripts can be found. For example,

```
setenv PPL_COMMAND_PATH .:/users/dwd/scripts
```

PPLUS will automatically search for scripts with the optional suffix ".ppc" after trying the script name without the suffix for each of the directories in the order listing in the command path. Note: pplus_home_directory/scripts is appended to this path.

```
setenv GRAPHTERM string
```

where "*string*" describes your graphics terminal and has the following allowed values:

VT240
TEK4010
TEK4105
HP2397
GVT+
MAC
TAB
GP220
ZENITH
TEK41XX
HIREZ
BW or **BLACK** (for use with a Sun Workstation)
NEW (for use with a Sun Workstation)

The entries that are used with suncore determine how a graphics window is opened. If **BW** or **BLACK** is included a single bit plane is used. If **NEW** is requested *PPLUS* will open a new window for graphics output. Both **NEW** and **BW** can be in the same GRAPHTERM string.

VAX/VMS

To get a copy of this manual

To get a copy of this manual, type the following lines on your terminal in response to the VAX/VMS prompt:

```
$ @DISK1:[OC.SYMBOLS]PLOT5
$ PPLUS_MANUAL
$ PPLUS_FONTS
```

The manual will be printed on the laser printer, and the *PPLUS* character fonts will be plotted on the Versatec plotter.

Required definitions

PPLUS requires several assignments and definitions to execute under VMS. The following should be included in your `LOGIN.COM` file (they must execute in both batch and in interactive mode). Then re-run your login.com, and you can use *PPLUS*:

```
$ @DISK1:[OC.SYMBOLS]PLOT5.COM
$ GRAPHTERM :== term_type,
```

where *term_type* describes your graphics terminal and has the following allowed values:

VT240
TEK4010
TEK4105
GVT+
MAC
TAB
ZENITH
TEK41XX

If you are using a VAX workstation monitor, it doesn't matter what you value you give the GRAPHTERM symbol, and you must use the *PPLUS* command **pltype** with an argument of either 3 or 4 to get interactive plots.

In order to provide automatic entry and exit into and out of graphics mode you should use the GRAPHTERM that corresponds to your terminal. If your terminal is a TEK4010 or TEK4014 compatible, but not one of the above, then place your terminal into graphics mode before plotting and use `GRAPHTERM :== TEK4010`. The execution of `PLOT5.COM` will define any other symbols needed by *PPLUS*.

PPLUS is entered interactively by typing `PPLUS` (or just `PPL`) in response to the VAX/VMS prompt.

Interactive help is available by typing `HELP PPLUS` in response to the VAX/VMS prompt. If you are in *PPLUS*, help is available by typing `HLP`.

Optional definitions

In addition to the above, the following VAX/VMS symbols and logicals may optionally be defined by the user:

PPL$RESET The "SAVE" file to be used by the *PPLUS* **reset** command (logical). Default is `PPL$EXE:PPL$RESET.DAT`

ECHO Defines the file to be used to echo *PPLUS* commands (logical). Default is `ECHO.DAT`.

PPL$STARTUP Defines an initialization or startup command file that will be executed each time *PPLUS* is entered (symbol). Default is no startup command file.

Example definitions:

```
DEFINE PPL$RESET DISK1:[userdir]reset.file
DEFINE ECHO your-echo.file
PPL$STARTUP :== DISK1:[userdir]startup.file}
```

3

Command Format

The Commands

The basic format for *PPLUS* commands is:

COMM[*:Q1:Q2 ...*][*,arg1,arg2,arg3 ...*][*,sarg1,sarg2 ...*]

where *COMM* is the PPL command. The numeric arguments (*arg1, arg2, ...*) may be numbers in any fortran format (e.g. 1.E-5, -6, 10.23) or blank. The character string arguments *sarg1, sarg2, ...* must begin with a non-numeric character string or be enclosed in quotes (“), i.e., `“100”`. If the numeric or character string arguments are blank, the input is considered null and the default is used. Where all numeric arguments are to be defaulted, they may be omitted entirely (i.e., blank entries need not be made).

PPLUS commands may have optional qualifiers (*Q1, Q2 ...*). The format for qualifiers is “*:value*” or “*:novalue*” for true or false, respectively.

All parameters must be separated by commas or blanks, except null entries which must have separating commas. Null entries are allowed except where noted in the specific command description.

Commands can be continued on sequential lines by inserting a “-” (minus sign) at the end of the line to be continued. Command lines may be up to a total of 255 characters long.

All commands/parameters may be entered upper or lower case. Conversion to upper case is performed automatically when required.

UNIX

The character used to designate that a qualifier follows is “:”. This character is referred to as the “qualifier escape character”.

VAX/VMS

The qualifier escape character used under VAX/VMS is “/” (solidus).

4 Command Synopsis

This is intended as a brief overview of the *PPLUS* commands. Commands are fully described in the Command Description chapter. Examples illustrating their use are in the Beginners Guide chapter.

Data Entry

These commands are used to extract the information from a file containing the data to be plotted.

rd	Reads/identifies file containing data to be plotted.
skp	Skips/identifies records on the data file being read.
rwd	Rewinds/identifies the data file.
format	Describes the format of the data file.
vars	Locates the data to be plotted in the records of the data file.
evar	Locates the data to be plotted in the records of the EPIC data file.
autolab	Controls automatic labeling of EPIC and BIBO data plots.

The following commands allow data entry from a source other than a file.

enter	Allows data entry from the keyboard.
linfit	Does a linear least squares fit on data already in a line and inserts the least squares line into the next available line.

PPLUS output files

echo	Controls echoing of *PPLUS* commands to a *PPLUS* echo file.
debug	Controls *PPLUS* debug mode (echoes after symbol substitution)
pltnme	Names the output plot file.
pltype	Controls the format of the output plot file.

PPLUS command files

@	Initiates reading of commands from a *PPLUS* command file.
echo	Controls echoing of *PPLUS* commands to a *PPLUS* echo file.
debug	Controls *PPLUS* debug mode (echoes after symbol substitution)

Axis

The following commands control axis labelling and appearance.

X- and Y-axis

xaxis	Controls numeric labeling and tics on the x-axis.
yaxis	Controls numeric labeling and tics on the y-axis.
axatic	Sets number of large tics automatically for x and y.
axlabp	Locates axis labels at top/bottom or left/right of plot.
axlen	Sets axis lengths.
axlint	Sets label interval for axes.
axlsze	Sets axis label heights.
axnmtc	Sets number of small tics between large tics on axes.
axnsig	Sets no. significant digits in numeric axis labels (auto only).
axset	Allows omission of plotting of any axis.
axtype	Sets axis type for x- and y-axis.
tics	Sets axis tic characteristics.
xfor	Sets format of x-axis numeric labels.
xfor	Sets format of y-axis numeric labels.
xlab	Sets label of x-axis.
ylab	Sets label of y-axis.

Time axis

time	Sets start and end of time axis, start time of data.
taxis	Sets time axis on, sets time series delta-t (minutes), orients axis.
txlabp	Establishes time axis label position (or absence).
txlint	Specifies which tics will be labeled.
txlsze	Sets height of time axis labels.
txnmtc	Sets number of small tics between large tics.
txtype	Sets type and style of time axis.

Labels

labs	Makes a moveable label (up to 25 labels allowed).
hlabs	Sets height of each moveable label.
rlabs	Sets angle for each moveable label.
labset	Sets character heights for labels.
llabs	Sets start position for a line to location of each moveable >label. Draws a line from the label to a point.
conpre	Sets prefix for contour labels (characters, color, font).
conpst	Set suffix for contour labels (characters, color, font).
title	Sets and clears main plot label (without making a plot).

xlab	Sets label of x-axis.
ylab	Sets label of y-axis.

Command Procedures

@	Initiates reading of commands from a *PPLUS* command file.
dec	Decrements a counter.
inc	Increments a counter.
if	Block **if** statement.
else	Block **if** statement.
endif	Block **if** statement.
while	**while** loop construct.
endw	**while** loop construct.
set	Sets the value of a *PPLUS* symbol.
show	Shows the value of a *PPLUS* symbol.
listsym	Lists values of defined *PPLUS* symbols.

Color and Fonts

Commands to change the pen number or the character font can be embedded in any labels character string. See the preceding section for label commands and the chapter on LABELS. See the following section on area fill plotting for color selection for area-filled contour plots.

@P*n*	Sets pen number "*n*" when embedded in a label.
@C*nnn*	Sets color to number "*nnn*" when embedded in a label.
pen	Sets pen number for each data line, axes and labels.
dfltfnt	Sets default character font for all labeling.
lev	Sets pen numbers (colors) for contour plots.

Color Area Fill Plots

The following commands control color area fill plotting.

area	Makes a smooth color area fill contour plot.
cbaxis	Sets color bar axis labeling.
cbcsze	Sets size of color bar axis labels.
cbfor	Sets format of color bar axis labels.
cblabp	Controls position of color bar on graph.
cblint	Controls labeling of tics on color bar axis.
colorbar	Makes a color bar for an area filled contour plot.
levels	Sets number of auto contour levels.
loadcmap	Loads a color map from a disk file.
sqfill	Sets area fill algorithm to square fill or triangle fill
pixel	Makes a color area fill contour plot using rectangles centered on the grid points.
fill	Makes a color area fill from a line.

Plot Appearance

The following commands control various aspects of the plot’s appearance.

origin Sets distance of plot origin from lower left corner of the box.
size Sets size of entire plotting region.
box Controls drawing of a box around the entire plotting region.
cross Controls drawing of lines through the point x=0, y=0 on graph.
line Sets characteristics for each X-Y plot line.
markh Sets character size for each X-Y plot line marks.
multplt Allows a composite of several plots (all kinds) on one page.
rotate Rotates plot by 90 degrees on screen and plotter.

Plot Generation

The following commands select the plot type and generate the plot.

plot Plots x-y pairs for all lines of data.
plotuv Makes stick plot of vector data for U,V pairs in line1.
plotv Makes stick plot of vector data for U in line1 and V in line2.
contour Makes contour plot.
view Makes a 3-D surface plot.
vpoint Sets the viewpoint for a 3-D surface plot.
vector Makes a plot of a vector field
multplt Allows a composite of several plots (all kinds) on one page.

Data Manipulation

linfit Does a linear least squares fit on data already in a line and inserts the least squares line into the next available line.
transxy Applies a linear transformation to variables x and y.
smooth Controls smoothing of contour type data.
limits Sets testing values for good data points.
window Controls windowing of data within axis bounds.

Map Transformations

mercat Transfrom using the Mercator projection.
peters Transform using the Peters projection.
polar Transform using the Polar Stereographic projection.

VAX/VMS specific commands

dir A listing is made of the current directory.
help VAX/VMS on-line help for *PPLUS*.
hlp Access on-line help from within *PPLUS*.

spawn Creates a VMS sub-process.

5 Beginners Guide

To use *PPLUS* a minimum of preparation is required. See the chapter on Getting Started for the symbol definitions that are required. Once this has been done *PPLUS* can be entered by typing `PPLUS` in response to the system prompt. The minimum number of commands needed to read in data and then plot the data are: **format** (sets the input format), **skp** (a command to position the file to a given record). **vars** (tells *PPLUS* how the data is arranged in each data record), **rd** (reads the data) and **plot** (create the plot).

To make contour plots, use the **vars** and the **rd** commands to read the data and then **contour**, **area** or **pixel** to create a contour plot. The name of the file containing the data can be specified with the **rd** or **skp** commands.

Multiple plots can be put on a page with the **multplt** command. Following are discussions of these commands and some examples of how these commands are used. For more information see the Command Description chapter.

Format

format informs *PPLUS* the type of the data file and the format the data has within this file. Valid formats include:

unf the data is unformatted (data type REAL)
free the data is formatted and in free form
(*xxx*) the data is formatted with a format of *xxx*, where *xxx* is a legal FORTRAN format, i.e., (3F10.2)

Vars

The next command you need to know about is **vars**. **vars** is a complicated command because it allows great flexibility in the organization of the data within each file record. Position of the characters 1, 2, and 3 within the command line indicates position of the X, Y, and Z variables within the data record. The format of the command is: **vars**, *ngrp*, *A1*, ..., *Ai* where *i* is the number of data values per data group

ngrp number of groups per record. For example, if the data file has Depth, Temperature pairs packed 3 pairs per record with a format of 3(F6.1,F6.2) then *ngrp*=3.

Aj 1, 2, 3 or blank to indicate that the variable in this position within the group is to be plotted as X (*Aj* = 1), Y (*Aj* = 2), Z (*Aj* = 3), or is not to be read at all (*Aj* = blank). An example will make this clearer.

EXAMPLE: `vars,1,,2,1`

First arg is 1	there is only 1 group per record (e.g. 1 scan per line of data) in the data file
Second arg is blank	Variable 1 in the data record is not to be read. (A1 = blank)
Third arg is 2	Variable 2 in the data record is to be plotted as Y (A2 = 2)
Fourth arg is 1	Variable 3 in the data record is to be plotted as X (A3 = 1)

No variable is to be read as Z.

The default is `vars,1,1,2` (i.e. one group per record, first variable is X, second is Y). The following are examples of the **vars** command:

`vars,1,,,1` tells *PPLUS* that there is one group of data per record and to read the third number in the record as the X variable. Since no Y variable location has been specified the Y variable will contain the sequence number.

`vars,5,1,2` lets *PPLUS* know that there are five groups of data pairs per record. Again the X variable is first and the Y second.

`vars,1,1,2,3` informs *PPLUS* that the data is X,Y,Z triplets with one group per record. The fact that X,Y, and Z appears tells *PPLUS* that the data is not on a regular grid and *PPLUS* should place it on an even grid. The method used to place the data on a regular grid and the grid itself are determined by the **rd** and **conset** commands.

`vars,1,,,,2,1` tells *PPLUS* that there is one group of data per record where the Y variable is the fourth number and the X the fifth number in the record.

`vars,1,3` tells *PPLUS* that there is one group of data per record and Z is the only variable in the group. This is for contour data which is already gridded. The **rd** command defines how the data is stored, i.e., which index varies fastest.

Skp and Rd

The name of the file containing the data to be plotted can be specified with either the **skp** or the **rd** command. The **skp** command tells *PPLUS* to skip records in the data file (e.g., header records or data which should not be plotted). Its format is **skp**,*n,file_name* where *n* is the number or records to skip, and *file_name* is the name of the data file and is an optional parameter. If the name of the data file is included, the data file will be rewound before skipping. If the data file name is omitted, the file will not be rewound before skipping.

The **rd** command informs *PPLUS* how many records to read and what file to read them from. If you are not making a contour plot, the format of the command is **rd**,*nx,file_name* where *nx* is the number of points to read from the data file and *file_name* is the name of the data file and is an optional parameter. If the data file name is included, the data file will be rewound before the data is read. If the data file name is omitted, the file will not be rewound before reading.

If you are making a contour plot, the **rd** command format is somewhat different. If Z is being read (a 3 in the **vars** command), **rd** defines the size of the plotting grid and prompts the user for the minimum and maximum values of X and Y to be used for the plotting grid. The format for **rd** is **rd**,*nx,ny,type,file_name* where *nx* and *ny* set the size of the grid for contour data read. Specifically, when X,Y,Z triplets are being read for contouring, the grid on which the data is plotted can be either coarser or finer or the same as the input data. If *nx*=50 and *ny*=21, then the data will be plotted on a grid which is 50 x 21 points (regardless of input data limits or gridding). *type* tells *PPLUS* whether the data is stored by rows (X varies fastest) or columns (Y varies fastest) if the data is already-gridded contour data. Finally, *file_name* is the data file name. If the data file name is included, the data file will be rewound before the data is read. If the data file name is omitted, the file will not be rewound before reading.

Plot and Contour

plot or **contour** initiates plotting. An optional label can be included and this label will be used to title the plot. The label must start with a non-numeric character. See following section on labels.

VAX/VMS examples

All the examples in this section can be typed in while running *PPLUS* interactively after typing *PPLUS* in response to the system prompt. The format of the following examples are valid under UNIX, however, the named directories do not exist. Just be sure you have first defined the *PPLUS* symbols according to the Getting Started chapter before you try to do this. Once the plot appears on your terminal, enter <CR> to exit from graphics mode and continue. To exit from *PPLUS*, type **exit** in response to the `ppl>` prompt.

Unformatted data, X-Y plot

The following example reads in data from an unformatted file with one group of data per record. The data to be plotted has X in the second position and Y in the first. The data file has 296 data points in it but we will read only 100 at a time. The data file also has an 8 record header that contains character data and must be skipped.

```
ppl>format unf
ppl>vars,1,2,1
ppl>skp,8,PPL$EXAMPLES:DEEP3000.AVG
ppl>rd,100
ppl>plot,The first 100 data points
ppl>rd,100
ppl>plot,The second 100 data points
```

Pre-gridded data, contour plot

The next example illustrates reading in data to be contoured. The data file is unformatted and does not have any header. The data is already gridded with 1 value of Z per record. Since only Z is read from the data file, the input grid and the plotting grid must be identical, and are specified by the **rd** command. The grid is 34 points in the x-direction and 5 points in the y-direction. The *PPLUS* RD command prompts for the minimum and maximum for the X-Y contouring grid. In this example, the grid is 34 points in the x-direction from 10 to -6.5 units and is 51 points in the y-direction from 0 to -500 units. *PPLUS* will read Z values from the data file assuming x varies fastest. This means that the Z values on the data file correspond to the following x, y pairs:

```
    x      y
 10.0      0
  9.5      0


  9.0      0
    .
    .
 -6.5      0
 10.0    -10
  9.5    -10
  9.0    -10
    .
    .
 -6.5    -10
    .
    .
```

```
ppl>format unf
ppl>vars,1,,3
ppl>rd,34,51,1,PPL$EXAMPLES:CTDDAT.DAT
ENTER XMIN,XMAX,YMIN,YMAX
rd>10,-6.5,0,-500
ppl>contour,A test plot for contouring
```

Ungridded data, contour plot

This example shows the reading in of ungridded contour data. The data is unformatted with Y,X,Z the order of the triplets. We define the grid for plotting to be 22 x 11 with X and Y limits of 1,22 and -.033,.0576. Although the data file contains less than 1000 points, we can give *PPLUS* a much larger number to read, and it will stop at the end-of-file without error.

```
ppl>format unf
ppl>vars,1,2,1,3
ppl>rd,22,11,PPL$EXAMPLES:GRIDWI.FMT
ENTER NUMBER PTS TO READ
rd>1000 ENTER XMIN,XMAX,YMIN,YMAX
rd>1,22,.033,.567
ppl>contour,An example of contouring with ungridded data
```

Time series plot

This example demonstrates the reading in of time series data and setting up the x axis to be a time axis. The data file contains a sequence number, which is the day of the year or Julian Day and temperature. Since the sequence number increments by 1 for 1 day, and delta-time is 1 day by default in *PPLUS*, there is no need to include the delta-time in the **taxis** command. The **taxis** command tells *PPLUS* that the time series has a delta-time of 1440 minutes (the default) and that the time axis is to be turned on. (The alternate form of the **taxis** command would be "`taxis,1440,on`".) The **time** command tells *PPLUS* that the time axis will start at 0000 1 Jul 1985, end at 0000 1 Dec 1985, and that a sequence number of 1 corresponds to a time of 1200 1 Jan 85. The **ylab** command sets the y-axis label. The **limits** command tells *PPLUS* to omit data where Y = 0. The **vars** command is not needed since the data is formatted with one group of data per record, with the X variable first and the Y variable second, which is the **vars** command default. The **cross** command suppresses the drawing of a solid line through x=0, y=0 on the plot. The **box** command suppresses the drawing of a box around the entire plotting region. The **skp** command names the data file and skips past the 5 header records at the beginning of the data file. The **rd** command reads the data. The **pltype** command sets the plotting medium to be both Tektronix compatible and binary suitable for routing to hardcopy devices. The **pltnme** sets the name of the output plot file. The **plot** command generates the plot. See the Command Description chapter for a full description of all *PPLUS* commands.

```
ppl>format (17x,f3.0,7x,f5.0)
ppl>taxis,on
ppl>time,W198507010000,W198512010000,W198501011200
ppl>ylab,Air Temperature
ppl>limits,0,yeq,on
ppl>cross,0
ppl>box,off
ppl>skp,5,ppl$examples:atlas.dat
ppl>rd
ppl>pltype,2
ppl>pltnme,atlas.plt
ppl>plot,ATLAS Air Temperature at 2N 165E
```

Additional examples are in the directory `PPL$EXAMPLES` in the form of *PPLUS* command files, which are the files with extension .PPC. Use the VAX/VMS command `DIR PPL$EXAMPLES:*.PPC` to see what the file names are. You can run these command files with the VAX/VMS command `PPLUS PPL$EXAMPLES:`*`xxx`*`.PPC`, where *`xxx`* is the name of the *PPLUS* command file. The file will generate a plot on your terminal. Enter a <CR> to exit from graphics mode and return to the VAX/VMS prompt. (Be sure that you have first defined the *PPLUS* symbols according to the Getting Started chapter before you do this.) See the chapter on Command Files for more information about using *PPLUS* command files. You can copy these *PPLUS* command files to your own directory with the VAX/VMS command `COPY PPL$EXAMPLES:*.PPC [ ]`. Then you can run them with the VAX/VMS command `PPLUS  xxx.PPC`, where `xxx` is the name of the *PPLUS* command file. You can experiment with *PPLUS* commands by editing the *PPLUS* command file to change the appearance of the plot, and then run *PPLUS* again with your new command file.

Multiple plot using MULTPLT

The **multplt** command allows the user to place several graphs on a single page. The graphs can be made with any of the plotting commands (e.g., **plot**, **plotv**, **contour**, **vector**). Size and spacing of the plots is controlled with the **multplt** command. This type of plot should not be the first attempted by the beginning user.

This segment of a *PPLUS* command file demonstrates the use of the **multplt** command to put 8 graphs on a single page (4 rows with 2 plots across in each row). Each graph is made with the **plot** command. The size and spacing of the plots on the page is controlled with the **multplt** arguments. The x- and y-axis lengths are 3.0 and 1.6 inches respectively. The space between the plots in the x- and y-directions is 1.5 and 0.5 respectively. The left and right borders are 1.25 inches wide and the top and bottom borders are 1.5 inches wide.

```
C Set up the MULTPLT arguments:
set yy 1.6
set xx 3.0
set xsp 1.0
set ysp 0.5
C Use the MULTPLT command:
multplt,2,4
‘xx’,’xx’
‘yy’,’yy’,’yy’,’yy’
1.25,’xsp’
‘ysp’,’ysp’,’ysp’,1.5
C Read and plot the data:
skp,3,DATA.DAT
while PPL$LINE_COUNT .le. 8 then
rd,100
plot
endw
```

Color area filled contour plots with AREA or PIXEL

The **area** and **pixel** commands produce color area filled plots. They can be used in place of the **contour** commands, and have the same arguments. These commands can be used on an X windows terminal or workstation. If you use them, use the **pltype** command to tell *PPLUS* that you are a workstation. If you don’t have an X workstation, you can still make the plots and then output them to a color area-fill hardcopy unit, such as a PostScript plotter. For this, be sure **pltype** is set to produce a binary metacode file.

You can make an area filled contour plot by simply replacing the **contour** statement with AREA or **pixel**. If you do this, you will get default level selection, color map and no color bar. The following are some fragments of *PPLUS* commands controlling area-filled contour plots. At the end of this section, there is a sample *PPLUS* command file which is complete, and demonstrates the entire process of reading data and producing an area-filled plot.

Examples of adding a color bar to an area filled plot.

```
c This segment of a .ppc file adds a color bar oriented
c horizontally .25 inches above the data plot.
c
set xpos 0
set ypos ‘ppl$ylen’ + .25
colorbar/nouser ‘xpos’, ‘ypos’, 0, ‘ppl$xlen’
area

c This segment of a .ppc file adds a color bar oriented
c vertically .25 inches to the right of the data plot.
c
set xpos ‘ppl$xlen’ + .25
set ypos 0
colorbar/nouser,’xpos’, ‘ypos’, 0, ‘ppl$ylen’
area
```

Example of loading a color map:

```
c This segment of a .ppc file adds a color bar oriented
c vertically .25 inches to the right of the data plot.
c
loadcmap,ppl$cmap:ljm.21
set xpos ‘ppl$xlen’ + .25
set ypos 0
colorbar/nouser,’xpos’, ‘ypos’, 0, ‘ppl$ylen’
area
```

Example of loading a color map and setting the number of levels equal to the number of points in the color map:

```
c This segment of a .ppc file adds a color bar oriented
c vertically .25 inches to the right of the data plot.
c It loads the color map ppl$cmap:ljm.21, containing 21
c colors. The “setclev” line sets the number of contour
c levels to 21 and the number of tics on the color bar axis
c to 7. Then it plots the contour lines over the area
c filled plot.
c
loadcmap,ppl$cmap:ljm.21
@ppl$util:setclev 21 7
set xpos ‘ppl$xlen’ + .25
set ypos 0
colorbar/nouser,’xpos’, ‘ypos’, 0, ‘ppl$ylen’
area/nowait
contour/overlay
```

Here is an annotated *PPLUS* command file which reads contour data, adds a color bar, sets parameters for the color bar axis and loads a color map.

```
C Plot type is binary and tektronix
pltype,0
C Specify format of sequential data file
format,unf
C Specify the desired contour levels
lev,(),(0,35,1,-3)(0,30,5,-1),dark(0,30,5)
C Read data to be contoured
vars,1,1,2,3
rd, 32, 51,1,170w.con
500000
-30.000, 1.000, 0., 500.
C
colorbar/nouser 0,4.65,0,’ppl$xlen’
cbaxis,8,29,1
cbfor,(i2)
loadcmap,ppl$cmap:breen_nogreen.109
area
```

6 Workstations

Several workstations are supported by *PPLUS*. User interaction for the placement of moveable labels is modified in a window environment. The user can specify centered, left justified or right justified text relative to the locator position. Optional lines can be drawn from the first locator position to the label position.

X11 Window System

The X11 window system can be used with a wide variety of workstations and windowing terminals. It is only necessary to link *PPLUS* with the X11 library and not any of the toolkits. The Makefile should be edited to include -DX11 in FFLAGS and CFLAGS. The presence of the DISPLAY environment variable lets *PPLUS* know that the X11 window system is available. You may use X11 and SunCore together in the same *PPLUS* executable, *PPLUS* will automatically choose X11 if the DISPLAY variable is set.

With X11 when a moveable label is requested without supplying the coordinates a single cross locator prompt will appear when the mouse cursor is in the graphics window. To select left justification, centering or right justification of the label with respect to the locator prompt click the left, middle or right mouse button, respectively. To select the start of a line with an arrow head click the left button while pressing the shift key and to select the start of a plain line click the middle button while pressing the shift key. If you have indicated that you want a line a locator prompt will appear again this time as a double cross, reposition the locator and click the mouse button that corresponds to the type of justification you want.

VaxStation II (GKS)

The GKS graphics system is used with the VaxStation II and VaxStation II/GPX when running VMS. To get a graph on the workstation monitor, you must use the command PLTYPE with an argument of either 3 or 4. When a moveable label is requested without giving coordinates a GKS window will be opened in which the user can chose to center, left justify or right justify the label. The user also has the option of specifying that a fancy (line with an arrow head) or a regular line will be drawn. After the selecting the desired option a locator prompt will appear. Position the locator and click any button. If you have selected one of the line types the window menu will appear again. You select one of the label justification types and position the locator.

Tektronix Terminal

When plotting is done, the cross hairs will come on if no X and Y position has been specified. Typing a C will center the label at the cross hairs or typing a R will position the label to the right of the cross hairs. By typing L or F then repositioning the cross hairs and then typing another character a line will be drawn from the first point to the second and the label will be drawn at the second point (if F was specified an arrow will be drawn). Any character other than L, F, R or C will cause the label to be drawn at the cross hairs.

7

Routing Plot Files

UNIX

In UNIX several programs are available to translate the meta file to a form ready for a specific graphics devices. All filters have a common subset of flags for rotating, centering and scaling the plot. Depending on the device there may be additional flags for controlling the output. The filters are:

m2gif [**-w** *width*] [**-h** *height*] [**-A** *area*] [**-a**] [**-i**] [**-b** *bsize*] [**-B** *bcolor*] [**-O** *out_file*] *file_name*

A filter that creates a gif output file.

-w *width*	width in pixels
-h *height*	height in pixels
-A *area*	area in pixels squared
-a	No autocrop
-i	interlace the gif
-b *bsize*	border size in pixels
-B *bcolor*	border color name or ‘red green blue’ rbg values.
-O *out_file*	output file name

m2hdf [**-w** *width*] [**-h** *height*] [**-A** *area*] [**-p** *pen_file*] [**-a**] [**-i**] [**-b** *bsize*] [**-B** *bcolor*] [**-O** *out_file*] *file_name*

A filter that creates a hdf animation output file.

-w *width*	width in pixels
-h *height*	height in pixels
-A *area*	area in pixels squared
-p *pen_file*	change the “pen” colors.

-a	No autocrop
-i	interlace the gif
-b *bsize*	border size in pixels
-B *bcolor*	border color name or ‘red green blue’ rbg values.
-O *out_file*	output file name

mpeg_encode

The Berkeley mpeg encoder has been extended to accept *PPLUS* meta files. See the documentation in the encoder directory for complete information about how to use this software with *PPLUS* meta files.

m2ps [**-C**] [**-R**] [**-P**] [**-s** *fctr*] [**-p** *pen_file*] [**-F**] [**-o** *orient*] [**-c** *cmodel*] [**-m**] [**-b** ‘*red green blue*’] *file_name* [*file_name ...*]

a filter that creates a postscript output file. See the UNIX man page for more information. where:

-C	don’t center the plot
-R	rotate the plot
-P	Scale to the size of the page.
-s *fctr*	scale the plot by *fctr*
-p *pen_file*	change the “pen” color
-F	Rotate and scale the plot to fill the page.
-o *orient*	Image the plot with **landscape** or **portrait** orientation.
-c *cmodel*	Use either **rgb** or **cmyk** color model. **rgb** is the default.
-m	Inhibit warning messages.
-b ‘*red green blue*’	Set the background color using the **rgb** specification.

m2xv [**-M** *colors*] [**-P** *print_config*] *file_name* [*file_name ...*]

a filter that creates a plot on a X11 workstation. See the UNIX man page for more information. where:

-M *colors*	Maximum number of colors
-P *print_config*	read printer configuration from *print_config*

VAX/VMS

Plot files and mom

PPLUS will create a device-independent binary plot file if the user issues the **pltype** command with an appropriate argument. *PPLUS* plot files are named `ZETA.PLT` by default (this can be changed with the **pltnme** command). A graphics post-processor called **MOM** is available to reformat these binary plot files and route them to a graphics device. **MOM** submits a batch job to `BETA$LOPRI` or `BETA$BATCH`. When the batch job has finished, the original plot files will have been renamed from `file.ext` to `file.`*`PLT_HHMMSS`*, and the plots queued to the appropriate device. A log file with the name `MOM_`*`HHMMSS`*`.LOG` is placed in the original directory when the **MOM** option **/LOG** is selected.

The command is (brackets [ ] enclose optional information):

MOM [*arg1* [*arg2* ...]]

The arguments for **MOM** are order independent and are separated by spaces. The arguments are:

[**F**[**ILE**]=]	file name (default `ZETA.PLT`)
[**D**[**EVICE**]=]	device (e.g. **TEK**, **VER** etc, default **VER**)
S[**CALE**]=	scale factor (default 1)
G[**RACE**]=	grace distance (inches, default = 0.25)
W[**IDTH**]=	width (paper width CAL only, default = 11.5)
C[**PLOT**]=	“cplot arguments” (CPLOT parameters CAL only, default=NULL)
[**NO**]**ROT**[**ATE**]	(rotate the plot, default **NOROT**)
[**NO**]**CEN**[**TER**]	(center the plot, default **CENTER**)
/[**NO**]**SAVE**	(save the input file, default **/SAVE**)
/[**NO**]**LOG**	(create a batch log file, default **/NOLOG**)
	/SMALL, **/LARGE** or **/TRAN**S (type of hard copy made, default **/SMAL**L)

File names which are the same as a legal device name (e.g. **VER**, **TEK**, etc.) are not allowed. The file name can contain any wild carding that is valid with the VAX/VMS rename command. The default file extension is `.PLT`.

Plotting devices

VER	Batch plot on Versetec V80 printer/plotter
TEK	Interactive plot on Tektronix compatible terminal
CPY	Batch plot on Tektronix 4691 hardcopy unit
CAL	Batch plot on CALCOMP plotter
HP	Batch plot on HP7550A plotter
PS	Batch plot on LN03R Postscript plotter
PS2	Batch plot on downstairs LN03R Postscript plotter
CVER	Batch plot on Versetec V2756 color plotter
CVERB	Batch plot on Versetec V2756 black/white plotter
LN03	Batch plot on TMAP1:: LN03 printer/plotter
HPT	Batch plot on TMAP1:: HP7475

Examples

- `$MOM ?` Will cause MOM to prompt for inputs. If the CPLOT argument is a ? you are then prompted for the CPLOT inputs.
- `$MOM CTD110W VER SCALE=1.25 ROTATE` Will instruct MOM to create a VERSATEC plot from the meta file CTD110W.PLT, rotate the plot 90 degrees on the paper and rescale the plot by a factor of 1.25.
- `$MOM CAL CPLOT=””` Will have MOM create a CALCOMP plot using ZETA.PLT and call CPLOT with the default parameters. If CPLOT is omitted then MOM will prompt for the CPLOT command line (omitting CCFILE).
- `$MOM TEMP.PLT;* CAL CPLOT=”/P1=BLK:.3”` Will cause MOM to send all the versions of TEMP.PLT to the CALCOMP with operator instructions to have pen 1 be black ink pen of 0.3 mm width.
- `$MOM HP *.MY\_PLOT;* /TRANS` Will send all plots with extension .MY_PLOT to the HP7550 plotter with operator instructions to plot on transparencies.

8 PPLUS Command Files

Introduction

PPLUS can be run using a *PPLUS* command file that contains the same commands used by *PPLUS* interactively. The file can have any name or extension, but the default extension is .ppc. To run a *PPLUS* command file named cmp.ppc, you can enter *PPLUS* by typing `PPLUS cmd.ppc` in response to the system prompt, or you can enter *PPLUS* in the usual way and give the *PPLUS* command @cmd.ppc. (See @ in the chapter on Command Description.)

Each time *PPLUS* is used, an echo file (named echo.file by default) is generated. This file can be edited (it should be renamed) with any text editor and used as a *PPLUS* command file in subsequent *PPLUS* sessions.

Symbol Substitution

PPLUS allows symbol substitution in a manner similar to VAX/VMS symbols. Global and local symbols are supported in conjunction with nested command files and parameter passing. The **set** and **show** commands create, modify and list the symbols. When initially entering *PPLUS* (i.e., at the first command level) the symbols are global and available to all command levels. At each subsequent command level, local symbols are created and used by default. Global symbols are used when no local symbol exists. If the symbol name is preceded by a star (*), the global symbol will be created, modified or substituted. Local symbols are only accessible at the current command level. (Each invocation of a command file or return from a command file changes the current command level.)

Parameters passed via the @ command line are named *P1, P2, P3, etc. ...* just as they are in VAX/VMS. Symbols are recognized by *PPLUS* by being enclosed by single quotes. Character strings can be enclosed in double quotes. For example:

```
set temp “This is a test label”
xlab ‘temp’
```

will have the same effect as:

```
xlab This is a test label
```

Several symbols are predefined. ‘**DATE**’ and `**TIME**’ and contain the current date and time. Date and time formats are dd-mmm-yy and hh:mm:ss for VMS and “mmm dd yyyy” and hh:mm:ss for UNIX. In addition, **P1** through **P*n*** are also predefined if the corresponding argument was passed via the @ command. For example, the command procedure `plotit.ppc` could be executed in *PPLUS* by typing `@PLOTIT 110W Temperature`. Then in the procedure `plotit.ppc`, the symbol `P1` will have the value “`110W`” and the symbol `P2` will have the value “`Temperature`”.

Symbols can also be defined and used in an array format, i.e., `` `P(3)’ `` will get symbol `P3` and `` `label(12)’ `` will access symbol `LABEL12`.

To have a single quote (‘) in the symbol or command line two single quotes must be used (‘ ’). To have a double quote (“) in the command line two double quotes (“ ”) are required.

Here is a sample *PPLUS* command file which demonstrates some of the new, powerful *PPLUS* features. In this example, the symbol `P1` has the value `110W`.

```
pltnme,’p1’.plt
format,(f5.0,15x,f15.0)
vars,1,1,2
skp,1,’p1’.dat
rd,60
debug,on
show p1
debug,off
plot,@TRMonthly data 1979-83 at ‘P1’ (‘date’ ‘time’)
```

The proceeding *PPLUS* command file (i.e. plotit.ppc) could be called repeatedly in *PPLUS* for different data files named `110W.DAT`, `140W.DAT,` etc. by entering the *PPLUS* commands `@PLOTIT 110W`, `@PLOTIT 140W`, etc. The resulting plot files, echo.file and graphs would be identified by the data file names of `110W`, `140W`, etc. The graph title will also include the time and date when the graph was made.

General Global Symbols

The global symbols set by *PPLUS* to allow information to be available in the command procedure are:

TABLE 1. General Global Symbols

Symbol	Command	Description
date		The current date dd-mmm-yy
ppl$command_file	@	The current command file name
ppl$eof	**rd**,**rwd**,**skp**	“YES” if and EOF was read
ppl$format	**format**	The current format
ppl$height	**size**	Height of the box
ppl$input_file	**rd**,**skp**,**rwd**	The current input file
ppl$lf_a	**linfit**	Constant from fit y=a+b*x
ppl$lf_a_stdev	**linfit**	Standard error of A
ppl$lf_b	**linfit**	Constant from fit
ppl$lf_b_stdev	**linfit**	Standard error of B
ppl$lf_r2	**linfit**	Regression coefficient squared
ppl$lf_res_var	**linfit**	Residual variance
ppl$lf_var	**linfit**	Total variance
ppl$line_count		The number of the last line read
ppl$pltnme	**pltnme**	The name of the plot file
ppl$pltnmext	**pltnmext**	The file extension of the plot file
ppl$points	**rd**	Number of data points in last line read
ppl$range_inc	**%range**	See Advanced Commands Chapter
ppl$range_high	**%range**	See Advanced Commands Chapter
ppl$range_low	**%range**	See Advanced Commands Chapter
ppl$teknme	**teknme**	The name of the tektronix file
ppl$view_x	**vpoint**	X viewpoint
ppl$view_y	**vpoint**	Y viewpoint
ppl$view_z	**vpoint**	Z viewpoint
ppl$width	**size**	Width of the box
ppl$xfact(*n*)	**transxy**	Xfact for line *n*
ppl$xlen	**axlen**	Length of X axis
ppl$xoff(*n*)	**transxy**	Xoff for line *n*
ppl$xorg	**origin**	Distance between origin and left edge
ppl$xfirst(*n*)		X value for the first data point in line *n*
ppl$xhigh	**rd**	The x coordinate for the maximum value in a contour grid

TABLE 1. General Global Symbols

Symbol	Command	Description
ppl$xlast(*n*)		X value for last data point in line *n*
ppl$xlow	**rd**	The x coordinate for the minimum value ina contour grid
ppl$xmax	**rd**	X max of contour grid
ppl$xmin	**rd**	X min of contour grid
ppl$xmax(*n*)		X max for valid data in line *n*
ppl$xmin(*n*)		X min for valid data in line *n*
ppl$yfact(*n*)	**transxy**	Yfact for line *n*
ppl$ylen	**axlen**	Length of Y axis
ppl$yoff(*n*)	**transxy**	Yoff for line *n*
ppl$yorg	**origin**	Distance between origin and bottom edge
ppl$yfirst(*n*)		Y value for the first data point in line *n*
ppl$yhigh	**rd**	The y coordinate for the maximum value in a contour grid
ppl$ylast(*n*)		Y value for last data point in line *n*
ppl$ylow	**rd**	The y coordinate for the minimum value ina contour grid
ppl$ymax	**rd**	Y max of contour grid
ppl$ymin	**rd**	Y min of contour grid
ppl$ymax(*n*)		Y max for valid data in line *n*
ppl$ymin(*n*)		Y min for valid data in line *n*
ppl$zmax		Z max for valid contour data
ppl$zmin		Z min for valid contour data
time		The current time hh:mm:ss

EPIC Global Symbols

The following global symbols are set by *PPLUS* when the **rd** command is executed and contain information from EPIC time series data headers:

TABLE 2. EPIC Time Series Symbols

Symbol	Description
ppl$epic_comment_data(*n*)	Data comment from header
ppl$epic_comment_first(*n*)	Data comment from header
ppl$epic_comment_second(*n*)	Data comment from header

TABLE 2. EPIC Time Series Symbols

Symbol	Description
ppl$epic_depth(*n*)	Depth of measurement
ppl$epic_descript(*n*)	EPIC series descriptor
ppl$epic_experiment(*n*)	Experiment identifier
ppl$epic_latitude(*n*)	Latitude
ppl$piec_longitude(*n*)	Longitude
ppl$epic_mooring(*n*)	Mooring identifier
ppl$epic_project(*n*)	Project identifier
ppl$epic_var_descript(n)(*n*)	Variable descriptor (header line2)
ppl$epic_xlab(*n*)	X-axis label
ppl$epic_ylab(*n*)	Y-axis label

The following global symbols set by *PPLUS* contain information from EPIC CTD data headers:

TABLE 3. EPIC CTD Symbols

Symbol	Description
ppl$epic_cast(*n*)	CTD Cruise and Cast identifier
ppl$epic_ctd1hd(*n*)	CTD First line of header
ppl$epic_ctd2hd(*n*)	CTD Second line of header
ppl$epic_ctd3hd(*n*)	CTD Third line of header
ppl$epic_ctd4hd(*n*)	CTD Fourth line of header
ppl$epic_comment_first(*n*)	Data comment from header
ppl$epic_comment_second(*n*)	Data comment from header
ppl$epic_date(*n*)	CTD Cast Date (GMT)
ppl$epic_latitude(*n*)	Latitude
ppl$epic_longitude(*n*)	Longitude
ppl$epic_xlab(*n*)	X-axis label
ppl$epic_ylab(*n*)	Y-axis label

The following global symbols set by *PPLUS* contain general EPIC information:

TABLE 4. EPIC General Symbols

Symbol	Description
ppl$epic_datafile(*n*)	EPIC data file for line *n*
ppl$input_file	EPIC/pointer file

PPLUS Global Symbols

The following global symbols are set by *PPLUS* when the **rd** command is used in conjunction with `format,pplus`. The following symbols are set from information stored in the *PPLUS* format DSF file headers. These symbols are:

TABLE 5. PPLUS DSF Global Symbols

Symbol	Description
ppl$p_comment_a	The first comment line
ppl$p_comment_b	The second comment line
ppl$p_comment_c	The third comment line
ppl$p_comment_d	The fourth comment line
ppl$p_comment_e	The fifth comment line
ppl$p_gorient	The orientation of the grid (degrees)
ppl$p_grid	Additional grid information
ppl$p_gtype	Grid type
ppl$p_lab	Main title of plot
ppl$p_pdate	Date of plot creation
ppl$p_pident	Identifier of plot file
ppl$p_psource	Source of plot file
ppl$p_ptime	Time of plot creation
ppl$p_user_a	First user field
ppl$p_user_b	Second user field
ppl$p_user_c	Third user field
ppl$p_user_d	Fourth user field
ppl$p_varname	Name of variable
ppl$p_vdate	Gregorian date field
ppl$p_vident	Variable identifier
ppl$p_vjdate	Modified julian date of variable
ppl$p_vlat	Latitude of variable (or grid)
ppl$p_vlong	Longitude of variable (or grid)
ppl$p_vmax	Maximum of variable
ppl$p_vmean	Mean of variable
ppl$p_vmin	Minimum of variable
ppl$p_vrms	Root mean square of variable
ppl$p_vstd	Standard deviation of variable
ppl$p_vtime	Time of variable
ppl$p_vvar	Variance of variable
ppl$p_xlab	X-axis label of plot
ppl$p_ylab	Y-axis label of plot

Command File Logic

There are several commands that enable the user to make command files more like small programs. These commands are similar to FORTRAN's block IF and C's WHILE loops. Commands have been introduced that enable the user to increment and decrement a counter stored in a symbol by one. In order to make command files more readable leading blanks and tabs are ignored.

The syntax for the *PPLUS* commands is given in the Command Description chapter.

Examples

In this example, *PPLUS* is exited when an end-of-file is encountered by the **rd** command. This illustrates both the block IF and the use of the global *PPLUS* symbol ppl$eof.

```
rd
if PPL$EOF .eq. "YES" then
   exit
endif
```

In the following example, the size of the plot is set to val by val inches if the value of the symbol val is less than or equal to 13 otherwise the size is set to 13 x 13.

```
if val .le. 13 then
   size 'val' 'bval'
else
   size 13 13
endif
```

In the next example, if `P1` is null then `P1` is set to temporary.plt and then the plot name is set to the value of the symbol `P1`.

```
if P1 .eq. "" then
   set P1 temporary.plt
endif
pltnme 'P1'
```

This **while** loop results in 10 plots of 100 points each from data file `DLDK1039.DAT`. (ppl$line_count is a *PPLUS* defined symbol for the sequence number of the last data line read.)

```
skp,DLKD1039.DAT
while PPL$LINE_COUNT .le. 10 then
   rd,100
   plot
endw
```

Arithmetic

Simple arithmetic can be performed using *PPLUS* symbols. The commands that perform these function are **set**, **inc** and **dec**. The **inc** and **dec** functions are primarily used to increment and decrement counters in **while** loops. The following **while** loop uses the counter to set the line type to a solid line for each line to be plotted (ppl$line_count is a *PPLUS* defined symbol for the number of the last data line read):

```
set count 1
while count .le. PPL$LINE_COUNT then
   line,'count',,0
   inc count
endw
```

The **set** command can be used to perform simple arithmetic on *PPLUS* symbols. The syntax for these arithmetic expressions have the form:

set *symbol num1 op num2*

where *op* is +, -, * or / (addition, subtraction, multiplication or division) and *num1* and *num2* are numbers. The numeric values must be separated from the operator *op* by spaces. The string will be used exactly as it appears if enclosed by double quotes ("). The following example centers a moveable label 0.5 inches above the top axis (ppl$xlen and ppl$ylen are *PPLUS* symbols for the X and Y axis lengths):

```
set xpos 'PPL$XLEN' / 2.0
set ypos 'PPL$YLEN' + 0.5
labs:nouser,1,'xpos','ypos',0,"A centered label"
```

Note: The qualifier escape character is "**/**" under VMS.

Symbol Arrays

As described in the Symbol Substitution section, *PPLUS* symbols can be defined and used as arrays. There are several general *PPLUS* global symbols which are defined as arrays, such as ppl$xlast(***n***) and ppl$ylast(*n*), the last x and y values for data line *n*. The array index, in parentheses, can be either a number or a *PPLUS* symbol. Examples will illustrate this. The following piece of a *PPLUS* command file uses moveable labels to write the line number to the right of the last point plotted for the last line read in. It uses the global *PPLUS* symbols ppl$xlast(*n*), ppl$ylast(*n*) and ppl$line_count.

```
set xpos 'PPL$XLAST(PPL$LINE_COUNT)'
set ypos 'PPL$YLAST(PPL$LINE_COUNT)'
labs 'PPL$LINE_COUNT','xpos','ypos',-1,'PPL$LINE_COUNT'
```

The array index can also be a user defined symbol. In the following example, the array `MON` contains the names of the first 3 months of the year. The graph title will be "`Daily Values for the Month of FEBRUARY`".

```
set mon(1) "JANUARY"
set mon(2) "FEBRUARY"
set mon(3) "MARCH"
.
.
.
set count 3
```

```
.
.
.
plot,”Daily Values for the Month of ‘mon(count)’”
```

The index of an array (inside parentheses) will be interpreted according to the following rules: 1) if it is a number, that number will be used as the array index, 2) if it is not a number, it will be interpreted as a symbol.

Special Functions

The functions described in this sections are all accessed with the **set** command. They can be accessed only with the **set** command. The functions enable string manipulation and formatting within *PPLUS* symbol values. There are also several special math functions and functions used to access *PPLUS* data that can be used with the **set** commands. The *PPLUS* functions are similar to some of the VAX/VMS lexical functions.

The general syntax is:

set *sym* **$***function (arg1, arg2,...)*

where “*sym*” is the symbol set by the function and “*function*” is the name of the *PPLUS* function. *PPLUS* functions and their arguments are described in the following sections. Where function arguments are indicated as symbols, they must be *PPLUS* symbols and cannot be strings. Where function arguments are indicated as strings, they can be enclosed in double quotes.

$EDIT

The command is:

set *sym_out* **$edit (***sym_in***,** *arg1* [*arg2 arg3...*] **)**

where:

sym_out symbol set by the function
sym_in symbol on which function is to work
arg1 **upcase** changes string in *sym_in* to upper case
trim trims leading and trailing blanks from *sym_in*
compress removes extra blanks from *sym_in* (reduces each group of blanks to a single blank)
collapse removes all blanks from *sym_in*

If multiple arguments are used, they can be separated by blanks, e.g., `set sym $edit(sym_in, upcase collapse)`. If commas are used as separators, the entire set of arguments must be enclosed in quotes, e.g., `set sym $edit(sym_in, ”upcase,collapse”)`.

For example,

```
set s1 “depth”
set s2 $edit(s1,upcase)
```

results in `S2` having the value “`DEPTH`”, and

```
set s1 “ depth ”
set s2 $edit(s1,upcase trim)
```

results in `S2` having the value “`DEPTH`”.

$EXTRACT

This function extracts selected characters from the input string. The first character in the string is in position 1. The command is:

set *sym_out* **$extract (***start***,***length***,***sym_in***)**

where:

sym_out	symbol set by the function
start	starting character position
length	length of character string to be extracted
sym_in	symbol on which function is to work

For example,

```
set s1 “February”
set s2 $extract(1,3,s1)
```

results in `S2` having the value “`Feb`”.

$INTEGER

This function converts a number to integer format. The command is:

set *sym_out* **$integer (***sym_in***)**

where:

sym_out	symbol set by the function
sym_in	symbol on which function is to work

In the following example, the symbol `MON` has been incremented, and will have the value “`2.00`”. The symbol `INT_MON` will have the value “`2`”.

```
set MON 1
.
.
.
inc MON
set INT_MON $integer(MON)
```

$LENGTH

This function returns the length of the input string. The command is:

set *sym_out* **$length (***sym_in***)**

where:

sym_out	symbol set by the function
sym_in	symbol on which function is to work

For example,

```
SET S1 “February”
SET S2 $LENGTH(S1)
```

results in S2 having the value “8”.

$LOCATE

This function locates a substring in the input string. The first character in the string is in position 1. The function returns zero if the string is not found. The command is:

set *sym_out* **$locate (***substrg***,***sym_in***)**

where:

sym_out	symbol set by the function
substrg	string to be located
sym_in	symbol function on which function is to work

For example,

```
set s1 “JAN 21,1987”
set s2 $locate(“,”,s1)
```

results in S2 having the value “7”.

$ELEMENT

This function extracts an element from an input string in which the elements are separated by a specified delimiter. The command is:

set *sym_out* **$element (***pos***,***delim***,***sym_in***)**

where:

sym_out	symbol set by the function
pos	position of element to be extracted
delim	delimiter
sym_in	symbol on which function is to work

For example,

```
set month “JAN/FEB/MAR/APR/MAY/JUN/JUL”
set mon $element(3,”/”,month)
```

results in `mon` having the value “`MAR`”, and

```
set month “JAN/FEB/MAR/APR/MAY/JUN/JUL”
set count 1
while count .le. 7 then
   set mon(count) $element(‘count’,”/”,month)
   inc count
endw
```

results in `MON(1)` = “`JAN`”, `MON(2)` = “`FEB`”, `MON(3)` = “`MAR`”, `MON(4)` = “`APR`”, `MON(5)` = “`MAY`”, `MON(6)` = “`JUN`”, `MON(7)` = “`JUL`”.

Math Functions

These functions are used to return the results of several mathematical functions.

The command is:

set *sym_out* **$***func* **(***args***)**

where:

$*func* one of the following mathematical functions

$sind(*x***)**	sine function (degrees)
$cosd(*x***)**	cosine function (degrees)
$asin(*x***)**	inverse sine function (degrees)
$acos(*x***)**	inverse cosine function (degrees)
$tand(*x***)**	tangent function (degrees)
$log(*x***)**	common logarithm
$ln(*x***)**	natural logarithm
$pow(*x***,***y***)**	raise x to the power of y
$sqrt(*x***)**	square root function
x	first argument (symbol or value to be evaluated)
y	second argument (symbol or value)

$GRID

This function returns the value of a data point in the grid at the specified coordinates.

The command is:

set *sym_out* **$grid (***i***,***j***)**

where:

i x coordinate index of the grid data
j y coordinate index of the grid data

$XVAL

This function returns the x value of a data point from the specified line at the given index.

The command is:

```
set sym_out $xval (i,l)
```

where:

i index value
l line number

$YVAL

This function returns the y value of a data point from the specified line at the given index.

The command is:

```
set sym_out $yval (i,l)
```

where:

i index value
l line number

9 Labels

Axis Labeling

Commands affecting the labeling of the axes are:

xaxis	Controls numeric labeling and tics on the x-axis.
yaxis	Controls numeric labeling and tics on the y-axis.
axatic	Sets number of large tics automatically for x and y.
axlabp	Locates axis labels at top/bottom or left/right of plot.
axlen	Sets axis lengths.
axlint	Sets label interval for axes.
axlsze	Sets axis label heights.
axnmtc	Sets number of small tics between large tics on axes.
axnsig	Sets no. significant digits in numeric axis labels (auto only).
axset	Allows omission of plotting of any axis.
axtype	Sets axis type for x- and y-axis.
xfor	Sets format of x-axis numeric labels.
yfor	Sets format of y-axis numeric labels.
xlab	Sets label of x-axis.
ylab	Sets label of y-axis.

The numeric axis labels are drawn such that zero will be labelled if it occurs between the low and high axis limits. If zero does not occur, then the first large tic (from the bottom or left) will be labelled. The large tics are forced to occur at integer multiples of the tic interval.

Embedded String Commands

Fonts

All labels in *PPLUS* can be plotted using any one of 21 character fonts and 12 symbol fonts. The default font is `SR` (Simplex Roman) and other fonts are called by preceding their two letter abbreviation by an @, i.e., `@CI` for complex italic. Symbol fonts are called by using the symbol number, i.e., `@MA01` plots the first symbol in MATH and `@MA12` will plot the twelfth symbol. Font changes (of the form @*XX*) can be embedded in any label string (e.g., **xlab**, **ylab**, **plot** commands).

@*font* selects "*font*" as the character or symbol font to be used, where the font abbreviations are listed below.

TABLE 6. Character Fonts

Code	Description
SR	Simplex Roman (default)
DR	Duplex Roman
TR	Triplex Roman
CR	Complex Roman
AS	ASCII Simplex roman
AC	ASCII Complex roman
CS	Complex Script
TI	Triplex Italic
GE	Gothic English
IR	Indexical complex Roman
SS	Simplex Script
CI	Complex Italic
II	Indexical complex Italic
SG	Simplex Greek
CG	Complex Greek
IG	Indexical complex Greek
GG	Gothic German
GI	Gothic Italian
CC	Complex Cyrillic
AR	Cartographic Roman
AG	Cartographic Greek

TABLE 7. Symbol Fonts

Code	Description
ZO	Zodiac
MU	Music
EL	Electrical
WE	Weather
MA	Math
SM	Simplex Math
MP	Map
LM	Large Math
IZ	Indexical Zodiac
IM	Indexical Math
CA	Cartographic
PM	Plot Marks

A clear font command **@CL** is available to change the default font. The next font called after a **@CL** becomes the new default font. *The font is reset to the default at the start of each label*. The command **dfltfnt** can also be used to change the default font to one of your choice.

Control characters (such as "_" for subscripting in *PPLUS* labels) for the two ASCII fonts AS and AC must be preceded by an **<ESC>** (this is ascii code=27). For example, to subscript while using the ASCII fonts you must have **<ESC>**_ in the label precede the character to subscript. Specifically, the *PPLUS* command "`plot,@ASZ<`**ESC**`>_5`" will result in a plot title of capital `Z` followed by a subscripted `5` in the ASCII Simplex Roman font. (See the section in this chapter on subscripting.)

Pen Selection

The pen may also be selected by giving the change pen command **@P***n*, where *n* is the character 1-9 and A-G. This allows the selection of up to 16 pens/colors. The color and font is reset to the default font and previous color after the character string is drawn. The PEN command can be used to change the default color by typing **pen,0,***default_color*. If you need to select a color index beyond the range of `P1` to `PG`, you can use the change color command **@C***nnn* where "*nnn*" is a 3-digit color index. (It must be 3 digits.)

Character Slant

The slant used in drawing the fonts may be changed by using the command **@Z***n*, where *n* is the character 0-9 and A-G. This allows the selection of slant angles from 0 to 45 in 16 increments. The slant is reset to zero after the character string is drawn.

Subscripting, Superscripting and Back Spacing

An **^** (up arrow) embedded in any label string will cause the next character to be drawn superscripted an _ (underscore) will draw it subscripted and a **** (backslash) backspaces over the last character drawn. The control characters **^**, _ and **** are available in the two ASCII fonts AS and AC by preceding the control character by an **<ESC>** (ASCII code=27). For example, to subscript while using the ASCII fonts you must have **<ESC>**_ in the label precede the character to subscript.

10 Data Formats

netCDF Files

Access to netCDF files is provided with the *nccalc* program. *Nccalc* uses the eps library that was designed to provide a uniform interface to several file formats. More information on the eps library is available at `ftp://ftp.pmel.noaa.gov/eps/README.eps`.

ASCII Files

The **format** to be used in reading from a sequential file is defined by the commands **format**, **vars**, and **rd**. Some definitions are useful:

NVAR	the number of variables per group
NGRP	the number of groups per record
NREC	the total number of records

For example, if the data consists of depth, u, v, t and the format is 8F10.2 (the format statement must be for an entire record) with two groups per record, the data would look like

D -- U -- V -- T -- D -- U -- V -- T

and *NGRP*=2, *NVAR*=4.

If you wanted to plot D as the Y variable, T as the X then, `format (8F10.2)` would be the correct **format** command and `vars,2,2,,,1` would be the correct **vars** command. (U and V are not read or plotted.)

However, if the format was F10.2,30X,2F10.2,30X,F10.2 then `format (F10.2,30X,2F10.2,30X,F10.2)` and `vars,2,2,1` would be appropriate.

If the data is *unformatted* the meanings of *NVAR* and *NGRP* are unchanged. Unformatted data is specified by the **format** command `format,unf`.

Reading will automatically stop at the end of the file and properly store the data.

EPIC Format Files

This is the standard format for data from the EPIC data base. The data files are binary sequential files with at least one header of 8 80-character lines followed by data records with 1 data scan per record. When the `format,epic` command is used, the file name specified with the **rd**, **skp** and **rwd** commands refers to the EPIC or pointer file. Variables to be read are specified with the **evar** command. Both time series EPIC data files and CTD EPIC data files are recognized by *PPLUS*. The **:ctd** qualifier on the **format** command tells *PPLUS* which type of EPIC data is being read.

DSF Files

BIBO Format

The BIBO data format consists of data files created using the DSF routines and a 145 word header in the BIBO format. This data format is in the standard dsf file format for data storage.

PPLUS Format

This is a data format that is produced using the DSF routines with the header and data in *PPLUS* format (`format,pplus`). The format must be followed to insure that *PPLUS* can read the header and data correctly. This format also contains information from which many useful *PPLUS* symbols are defined automatically when the file is read. See your system manager for more details in order to create your own files.

DSF Format

This data format is that produced by the DSF routines with the header and data in *PPLUS* format. The format must be followed to insure that *PPLUS* can interpret the data file read correctly.

A single data file consists of a single header record and any number of data records followed by an EOF. The header must be either an array or other sequentially organized data set of 38 real variables. Below is the expected format.

TABLE 8. DSF Format Header

Integer Position	Word	Description
1	XPTS	first four created by CLSDSF
3	ZMIN	
5	ZMAX	
7	ZMEAN	

TABLE 8. DSF Format Header

Integer Position	Word	Description
9	XMIN	minimum x value (real)
11	XMAX	maximum x value (real)
13	KX	number of x grid points (int*4)
15	YMIN	minimum y value (real)
17	YMAX	maximum y value (real)
19	KY	number of y grid points (int*4)
21	ITYPE	data type 0=2-d set, 1= 1-d set (int*4)
23-38	LAB(16)	main label hollerith (int*2)
39	NCH	number of characters in LAB (int*4)
41-56	IXLAB(16)	x axis label hollerith (int*2)
57	NXLB	number of characters in IXLAB (int*4)
59-74	IYLAB(16)	y axis label hollerith (int*2)
75	NYLB	number of characters in IYLAB (int*4)

All labels use SYMBEL to generate the plotted characters. The labels are optional, but if not used they should contain blanks.

ITYPE=0

Data must be stored in a linear array as:

Z(1,1), Z(2,1), ... , Z(KX,1), Z(1,2), ... , Z(KX,KY)

or as a 2-d array where the array is dimensioned as KX,KY.

Assuming the following arrays exist, ITYPE=0 data can be created as follows: HEAD(38),Z(25,50) NOTE: use EQUIVALENCE to set the integers in the real array.

```
CALL OPNDSF(file_name,’WR’,ILUN)
CALL WRHDSF(ILUN,38,HEAD)
CALL WRDDSF(ILUN,1250,Z)
CALL CLSDSF(ILUN)
```

where file_name is the file name and ILUN is the logical unit to be used.

ITYPE=1

Data must be stored as a linear array as:

X(1), X(2), ... , X(KX), Y(1), Y(2), ..., Y(KX)

in this case KX= length of the series and KY must be set to 1, there must be KX of each X and Y in the data set. Given, HEAD(38),X(200),Y(200) KX=100 then,

```
CALL OPNDSF(file_name,’WR’,ILUN)
CALL WRHDSF(ILUN,38,HEAD)
CALL WRDDSF(ILUN,KX,X)
CALL WRDDSF(ILUN,KX,Y)
CALL CLSDSF(ILUN)
```

where KX is the number of pairs. The DSF routines are available in a user library by Task building with PMEL:[PPLUS.V1_1.SOURCE]OURLIB/LIB.

11 Subroutine Calls

PPLUS can be run from another program by linking the *PPLUS* libraries and using the following subroutines. When *PPLUS* is run via the subroutine calls the behavior is identical to the stand alone version. See the fortran source code of `pplus.F` or `PPLUS.FOR` for an example of how these subroutines are used.

PPLUS routines

opnppl is used to initialize the *PPLUS* package. The logical unit numbers specified should be chosen so that *PPLUS* will not interfere with the controlling programs input/output. **opnppl** must be called before and *PPLUS* routines are used.

opnppl subroutine opnppl(efile,elun,dlun,mlun,clun,ltt,key1,key2,epl1,epl2)
character efile*(*)
integer elun,dlun,mlun,clun,ltt,key1,key2,epl1,epl2

efile	echo file name
elun	echo lun
dlun	data lun
mlun	multplt temporary file lun
clun	command file lun
ltt	terminal lun
key1	first key file lun
key2	second key file lun
epl1	first EPIC lun
epl2	second EPIC lun

pplcmd runs *PPLUS* in one of three modes: 1) command input is from a command file specified by *fromi* and the command file arguments are given in *linei*; 2) command input is from the terminal, *fromi* is the device name from the terminal; 3) command input is given is *combuf* of length *icmsze*. Whichever method you use the unused variable must be either zero or set blank. Control will be returned to the calling program when *PPLUS* processes either all the commands in the buffer have been executed, a return or EOF from the command file is processed or the exit command is issued.

pplcmd

```
subroutine pplcmd(fromi,linei,isi,combuf,icmdim,icmsze)
character fromi*(*),linei*(*),combuf(icmdim)*(*)
integer isi,icmdim,icmsze
```

fromi	command file or device (/dev/tty or TT: for terminal)
linei	input line for parameters
isi	length of line in characters
combuf()	command buffer
icmdim	dimensioned length of combuf
icmsze	number of lines in combuf

pplldc is used to load contour data directly from memory into *PPLUS* without going through the **rd** command. The meaning of the parameters is consistent with the **rd** command.

pplldc

```
subroutine pplldc(k,z,mx,my,imn,imx,jmn,jmx,pi,pj,nx,ny,xmin,ymin,dx,dy,tstrt,xdt)
integer k,mx,my,imn,imx,jmn,jmx,nx,ny
real pi(*),pj(*),z(mx,my),xmin,ymin,dx,dy,xdt
character tstrt*12
```

k	= 0 already on an equally spaced rectangular grid
	= 1 on an unequally spaced, but rectangular grid
z	input array
mx	dimensioned x size of z(mx,my)
my	dimensioned y size of z
imn	initial x index value
imx	final x index value
jmn	initial y index value
jmx	final y index value
pi	x positions pi(mx)
pj	y positions pj(my)
nx	number of x grid points in output buffer
ny	number of y grid points in output buffer
xmin	x position of (1,1) in output grid
ymin	y position of (1,1) in output grid
dx	spacing of x grid points in output grid
dy	spacing of y grid points in output grid
tstrt	start time in WHOI format, corresponds to xt=1.0
xdt	sample rate in minutes for x

pplldv is used to load the second component of gridded data directly from memory into *PPLUS*. The meaning of the parameters is consistent with the **rd** command. The **pplldv** routine is only to be used immediately after **pplldc** has been used, **pplldv** assumes the same grid definitions from the **pplldc** routine. WHOI date format is either Wyymmddhhmm or Wyyyymmddhhmm.

pplldv

```
subroutine pplldv(k,z,mx,my,imn,imx,jmn,jmx)
integer k,mx,my,imn,imx,jmn,jmx
```

```
real z(mx,my)
```

k	= 0 already on an equally spaced rectangular grid
	= 1 on an unequally spaced, but rectangular grid
z	input array mx -- dimensioned x size of z(mx,my)
my	dimensioned y size of z
imn	initial x index value
imx	final x index value
jmn	initial y index value
jmx	final y index value

pplldx allows the calling program to load x-y pairs of data directly into the *PPLUS* plotting buffers from memory. *tstrt* and *xdt* are used only for time series data. The parameters have meanings similar to the **rd** and **vars** commands.

pplldx

```
subroutine pplldx(icode,xt,yt,npts,tstrt,xdt)
integer icode,npts
real xt(*),yt(*),xdt
character tstrt*12
```

icode	= 0 use both x and y
	= 1 use x only
	= 2 use y only
xt	x data
yt	y data
npts	number of xt and yt points
tstrt	start time in WHOI format, corresponds to xt=1.0
xdt	sample rate in minutes for x

tstrt and xdt are used only for `taxis,on`

clsppl is called when the calling program is done with *PPLUS*. This routine must be called to insure that all the temporary buffers are flushed and temporary files are removed.

clsppl

```
subroutine clsppl
```

PPLUSR routine

pplusr is a subroutine that is called by *PPLUS* when ever the **usr** command is issued. **pplusr** should not directly access any of the *PPLUS* common blocks or variables. All interaction should be via the routines described in this chapter.

pplusr

```
subroutine pplusr(command,ier,msg)
character command*255,msg*80
integer ier
```

command	command passed to pplusr from *PPLUS*
ier	= 0 no error
	= 1 error

	=-1 non fatal error
msg	a string containing a description of the error

Symbol routines

The following routines are used to store, read and delete *PPLUS* symbols. *PPLUS* symbols are stored in upper case. The naming convention for global symbols applies to these routines.

putsym is used to create or update a symbol with a new value. These symbols will be available to *PPLUS* command files.

putsym

```
subroutine putsym(sym,value,nc,ier)
character sym*30,value*255
integer nc,ier
```

sym	symbol name‘
value	value of symbol
nc	length of value
ier	= 0 success
	= 1 error

getsym is used to access the symbols created by the calling program, the users command file or *PPLUS*.

getsym

```
subroutine getsym(sym,value,nc,ier)
character sym*30,value*255
integer nc,ier
```

sym	symbol name
value	value of symbol
nc	length of value
ier	= 0 success
	= 1 symbol not found

delsym is used to delete symbols from *PPLUS*.

delsym

```
subroutine delsym(sym,ier)
character sym*30
integer ier
```

sym	symbol name
ier	= 0 success
	= 1 symbol not found

PPLUS format routines

Here is a group of six routines used to load data into and retrieve data from a header in the *PPLUS* formatted DSF file header. (See the chapter on data formats for more information.) The argument list for the subroutines that load and get the header information are identical for each type of data.

ldgrid and **gtgrid** load and get information about the data grid.

ldgrid

```
subroutine ldgrid(xmin,xmax,nx,ymin,ymax,ny,itype,ang,gty,info)
real xmin,xmax,ymin,ymax,ang
integer nx,ny,itype
character gty*8,info*32
```

xmin	minimum x value (user units)
xmax	maximum x value (user units)
nx	number of x grid points
ymin	minimum y value (user units)
ymax	maximum y value (user units)
ny	number of y grid points
itype	data type (=0 2-d set, =1 1-d set)
ang	grid orientation angle
gty	grid type
info	additional grid information

ldcmnt and **gtcmnt** load and get user comment information.

ldcmnt

```
subroutine ldcmnt(cmnta,cmntb,cmntc,cmntd,cmnte)
character*120 cmnta,cmntb,cmntc,cmntd,cmnte
```

cmnta	user supplied comment
cmntb	
cmntc	
cmntd	
cmnte	

lduser and **gtuser** load and get user fields.

lduser

```
subroutine lduser(usera,userb,userc,userd)
character*32 usera,userb,userc,userd
```

usera	user supplied field
userb	
userc	
userd	

ldplid and **gtplid** load and get plot identification information.

ldplid

```
subroutine ldplid(ident,source,date,time)
character ident*32,source*32,date*16,time*16
```

ident	identifier of plot file
source	source of plot file
date	date of plot file creation
time	time of plot file creation

ldvar and **gtvar** load and get information about the data.

ldvar subroutine ldvar(name,date,time,ident,mean,var,std,rms,min,max,jdate,lat,long)
character name*32,date*16,time*16,ident*32
real mean,var,std,rms,min,max,jdate,lat,long

name	name of variable
date	gregorian date
time	time of data
ident	variable identification
mean	mean of variable
var	variance of the data
std	standard deviation of the data
rms	root mean square of variable
min	minimum of variable
max	maximum of variable
jdate	julian date of variable
lat	latitude of variable (or grid)
long	longitude of variable (or grid)

ldaxlb and **gtaxlb** load and get the axis labels.

ldaxlb subroutine ldaxlb(xlabel,ylabel,main)
character xlabel*80,ylabel*80,main*120

xlabel	x axis label
ylabel	y axis label
main	main title of plot

More information about the *PPLUS* data format can be found in the ppl.inc include file. This file defines the header information. There is information in the header not directly available via the above routines.

12 Command Description

! *unix command*

Spawns a unix shell to execute the indicated command. Control is returned to *PPLUS* immediately after the command is completed.

@file_name *:qualifier arg1 arg2 arg3 ...*

Reads commands from the file file_name until an EOF, blank line, a **return** command is executed or the file ends, then reverts to the previous command level for input. Default extension is ".ppc". The current command file name is placed in global symbol ppl$command_file.

PPLUS can be started with a command file specified by typing "`$PPL  file_name`", where *file_name* is the command file name. *PPLUS* will produce no screen output if called from a BATCH file. *PPLUS* will terminate and not pass control back to the SYS$INPUT file.

The arguments may be any legal string. The arguments arg1,arg2,etc are **set** to the local symbols P1, P2, etc. For example:

```
@command_file your_file “A label” “PLTYPE 2”
```

The local symbols will be:

```
P1 = your_file
P2 = A label
P3 = PLTYPE 2
```

These symbols can then be substituted into the command file. *Qualifiers* are (default in parenthesis):

:[no]echo Controls echoing to the file echo.dat during execution. (**:noecho**)

:[no]debug Sets **debug** mode during execution. In debug mode the commands are written to the echo file after symbol substitution has occurred. (**:nodebug**)

:[no]quiet Turns off messages to the terminal. (**:noquiet**)

:[no]log Echos commands to terminal. (**:nolog**)

:[no]latch Causes the current *qualifiers* to be the new default for all command levels. (**:nolatch**)

area *:qualifier vcomp, angle, label*

Does an area-filled contour plot of data in buffer. **Area** produces a smooth area filled plot. **Pixel** puts an area filled rectangle at each grid point, centered on the grid point. **Area** and **pixel** are like the **contour** command except **contour** draws contour lines instead of producing an area filled plot. Use **colorbar** to put a colorbar on the graph. Other commands related to area filled contour plots are **cbaxis**, **cbcsze**, **cbfor**, **cblabp**, **cblint**, **loadcmap**, **levels**, **lev** and **sqfill**. See the section on Examples for sample commands for area filled plots.

The label will replace that in the current main label buffer. *label* is optional. If either axis is log that index must be equally spaced in log-space. (i.e. 10**(xmin+dx)) **pixel** does not take the log of the coordinate. The label cannot begin with a numeric character, i.e., 95W. You can plot a number by specifying a font, e.g., @SR100 meters.

vcomp indicates which vector component to contour. Default is 1. *Vcomp* is to be used when a vector field has been read in. See the **vecset** and **vector** commands.

angle The angle used to rotate the contour plot (counter-clockwise) relative to the X and Y axes (degrees).

Valid *qualifiers* are:

:[no]wait Controls whether *PPLUS* pauses after plot completion. Pause is signaled by a tone and terminated by typing a character. If an <ESC> is typed *PPLUS* will return from the current command level to the lowest command level. Default = **:wait**.

:[no]overlay Controls whether *PPLUS* overlays this plot on the preceding plot. The axes and their labels are not redrawn. Moveable labels (**labs** command) will redraw. The default is **:nooverlay** which causes this plot to be a new plot.

auto { **on** | **off** }

Turns **on** and **off** the automatic copying of plots while at a TEK terminal. Default=**off**

autolab { **on** | **off** }

On (default for BIBO and EPIC data) to get graph labels from data file headers. **Off** (default for other data formats) for manual entry of graph labels. Default=**off**

axatic *aticx, aticy*

Sets the number of large tics in auto mode for X and Y axes. Default=5

axlabp *labx, laby*

Sets the numeric and character label position for X and Y axes. -1=bottom/left of plot, 0=no label, +1=top/right of plot. Default=-1

axlen *xlen, ylen*

Sets the X and Y axes length in inches. *Xlen* is also used as the length in inches of the time axis. Default=5.5,4.0 The values of *xlen* and *ylen* are placed in global symbols ppl$xlen and ppl$ylen.

axlint *lintx, linty*

Sets the label interval for X and Y axes. Labels are only drawn for large tics. Default=2, i.e. every other large tic.

axlsze *hgtx, hgty*

Sets the numeric label height for X and Y axes in inches. Default=0.10 If *hgtx* or *hgty* is negative the numeric axis labels are multiplied by -1 before plotting. See **labset** for character labels.

axnmtc *nmtcx, nmtcy*

Sets the number of small tics between large tics for X and Y axes. Default=0

axnsig *nsigx, nsigy*

Sets the number of significant digits in labels for auto labelling. Default=2

axset *top, bot, left, right*

Sets the flags controlling the plotting of the four axes. If =1 axis is visible, =0 axis is invisible. The default for all axes is visible.

axtype *typex, typey*

Sets the axis type for X and Y axes. 1 - normal, 2 - log, 3 - inv-log. Type 3 axis draws the top/right axis inverse and the bottom/left normal. Default=1

baud *ib*

Sets baud rate. Null entry not allowed.

ib	Baud rate, default=110

box { **on** | **off** }

Turns on and off the box that is drawn around the entire plotting region. Default is **on**.

c

Comment. This command can be used to comment your @ files. No action is done when this command is processed. The **c** must be followed by at least *one* blank space.

cbaxis *xlo, xhi, xtic*

Sets the color bar axis for area-filled contour plots made with **area** or **pixel**. If argument list is blank, the color bar axis labels are cleared. Default is auto scale.

xlo	axis minimum (beginning of axis)
xhi	axis maximum (end of axis)
xtic	distance between tics

cbfor *frmt*

Sets the format for the color bar axis label.

frmt 0 or (a format), default=0 (auto label) To create an integer numeric label the format must begin as “`(I`” or “`(i`”.

cblabp *n*

Specifies color bar axis label position. Use -1 for an axis below a horizontal color bar or to left of a vertical color bar, 0 for no label, or +1 for an axis above a horizontal color bar or to the right of a vertical color bar. (default is +1)

cblint *label_int*

Sets the label interval for X or Y axis of the color bar. If **cblabp** and **cblint** are set to zero no axis will be drawn on the colorbar.

label_int labeling interval for tics on color bar axis. (e.g. 0 for no labels, 1 for every tic, 2 for every other tic) (default=2)

cbcsze *ht*

Specifies height of color bar axis labels (inches). (default is .1)

colorbar *xpos,ypos,orient,hgt,width*

Draws a color bar to accompany an area filled contour plot made with the **area** or **pixel** commands. Other commands related to area filled contour plots are **cbaxis**, **cbcsze**, **cbfor**, **cblabp**, **cblint**, **loadcmap**, **levels**, and **lev**. There are examples of the use of **colorbar** in the section on Examples.

xpos,ypos x and y position in user units or in inches (no defaults)
orient orientation. Use *orient*=1 for vertical color bar and *orient* = 0 for horizontal color bar (default is 1)
hgt height of color bar in inches (default 4)
width width of color bar in inches (default .25)

Valid *qualifiers* are:

:[no]user determines units of x and y positions. Default is **:user**. If **:nouser** units are inches from the origin. (see the **origin** command)

clsplt

Closes the metacode file. Not to be confused with **%clsplt**, which is documented in the Advanced Commands Chapter.

conpre *prefix*

Sets a prefix string for the numeric contour labels of up to 10 characters. For example, “`CONPRE,@P2@TR`” will give labels using pen 2 and triplex roman font. Default = blank.

conpst *postfix*

As in **conpre,** but sets up to 10 characters following the contour numeric label. For example, “`CONPST,cm/sec`” will give contour labels like “10 cm/sec”. Default = blank.

conset *hgt, nsig, narc, dashln, spacln, cay, nrng, dslab*

Sets parameters for contouring and placing random data on a grid. Must be issued before the **rd** command.

hgt height of contour labels, default=.08 inches

nsig no. of significant digits in contour labels, default=2

narc number of line segments to use to connect contour points, default=1

dashln dash length of dashes mode, default=.04 inches

spacln space length of dashes mode, default=.04 inches

cay is the interpolation scheme. If *cay*=0.0, Laplacian interpolation is used. The resulting surface tends to have rather sharp peaks and dips at the data points (like a tent with poles pushed up into it). There is no chance of spurious peaks appearing. As *cay* is increased, Spline interpolation predominates over the Laplacian, and the surface passes through the data points more smoothly. The possibility of spurious peaks increases with *cay*. *cay* = infinity is pure Spline interpolation. An over relaxation process in used to perform the interpolation. A value of *cay* = 5.0 (the default) often gives a good surface.

nrng Any grid points farther than *nrng* away from the nearest data point will be set to “undefined” (1.0E35). Default=5

dslab nominal distance between labels on a contour line. Default = 5.0 inches

contour *:qualifier vcomp, angle, label*

Does a contour plot of data in buffer. *Label* will replace that in the current main label buffer. *Label* is optional. If either axis is log that index must be equally spaced in log-space (i.e. 10**(xmin+dx)). **Contour** does not take the log of the coordinate. The contour lines will be plotted with the pen selected for line 1. See the **area** and **pixel** commands for color area fill contour plots. The label cannot begin with a numeric character, i.e., 95W. You can plot a number by specifying a font, e.g., @SR100 meters.

vcomp indicates which vector component to contour. Default is 1. *Vcomp* is to be used when a vector field has been read in. See the **vecset** and **vector** commands.

angle The angle used to rotate the contour plot (counter-clockwise) relative to the X and Y axes (degrees).

Valid *qualifiers* are:

:[no]wait Controls whether *PPLUS* pauses after plot completion. Pause is signaled by a tone and terminated by typing a character. If an <ESC> is typed *PPLUS* will return from the current command level to the lowest command level. Default = **:wait**.

:[no]overlay Controls whether *PPLUS* overlays this plot on the preceding plot. The axes and their labels are not redrawn. Moveable labels (**labs** command) will redraw. The default is **:nooverlay** which causes this plot to be a new plot.

cross *icode*

Turns on and off the drawing of a solid line through (0,0) on a plot. Optionally can draw vertical and horizontal lines. Draws line through (xoff,yoff) when either **transxy** or **line** command is used to apply a transformation to the data.

icode 0 cross off (default)
1 draw through (0,0)
2 horizontal line through each yoff

3 vertical line through each xoff
4 horizontal and vertical through each xoff, yoff

datpt *type, mark*

Controls the drawing of marks on a contour plot along the x and/or y axis on a grid at the points where the raw ungridded X,Y,Z triplets are located.

type 0 no points drawn (default)
1 points drawn along the x axis
2 points drawn along the y axis
3 points drawn at each raw input value
mark 0 use the default mark (default) other use the specified mark to denote the location.

The default mark is down arrow for x axis, left arrow for y axis, and pluses for type=3. (also see **markh**)

debug { **on** | **off** }

Turns on and off the debugging mode. In debug mode the input lines are echoed to the echo.file file after symbol substitution. Default = **off**.

dec *symbol*

Decrements the value stored in *symbol* by one. If *symbol* does not exist it is created and given a value of zero.

delete *symbol*

Deletes *symbol* from the symbol table.

dfltfnt *font*

Sets the default font used for all labelling. *PPLUS* initially uses Simplex Roman (SR) as the default font. Fonts are still selectable using the font command @xx, where xx is the two letter font code. NOTE: This command also replaces the string set by the **conpre** command with the selected font. The default font is not saved with **multplt**. This command changes the environment and can only be changed back with another **dlftfnt** command or using the @CL command.

font the two letter code for the new default font (no default)

dir *arg*

Prints a listing of files with names or extensions that match *arg*.

e *nccalc_command*

These commands allow algebraic manipulations and data editing with EPIC Data System files and netCDF formatted files. The **e** command calls the *nccalc* program to execute *nccalc_command*. See the *nccalc* manual for more information.

echo { **on** | **off** }

Turns on/off echoing of *PPLUS* commands in the echo file “echo.file”. Default is **on**. In the VAX/VMS environment ECHO is a logical that can be defined prior to entering *PPLUS* (e.g., DEFINE ECHO echo_file.echo). Default is for echoing to go into the file echo.file.

english

Sets the internal conversion factors in Complot to inches. This is the default condition. (see the **metric** command)

enter

Allows the input of X,Y pairs from the terminal. *PPLUS* prompts the user with `enter>`. Type **end** to stop.

evar *:qualifier x-var, y-var*

Specifies which EPIC variables are to be plotted as x and y when “`format,EPIC`” command has been given. The EPIC/pointer file is named with the **rd** command, and each call to **rd** results in reading another EPIC data file as indicated by the EPIC/pointer file. *PPLUS* can extract axis labels and a plot title from the data file headers. Use “`format:CTD,EPIC`” to tell *PPLUS* that EPIC CTD data is being read. Use “`format,EPIC`” to tell *PPLUS* that EPIC time series data is being read. See **format** command description for all the EPIC defaults.

x-var	Variable to be plotted as x
y-var	Variable to be plotted as y

“`evar ?`” displays a list of variables possible for *x-var* and *y-var.*

Examples of variables are TIM (time), U (zonal velocity), V (meridional velocity), etc. If you want to plot x=time and y=zonal velocity, the command would be “`evar,tim,u`”.

Variables can also be specified by either column number within the data file or by EPICKEY variable code. To specify the column number of the variable in the EPIC data file use Cn for either *x-var* or *y-var* where n is the column number. For example, EPIC current meter data files generally have variables DATE, TIME, U, V, SPEED, DIRECTION, and the command to plot x=time and y=speed is “`evar,TIM,C5`”.

To specify the variable by EPICKEY variable code, use En for either *x-var* or *y-var* where n is the EPICKEY variable code. For example, an EPIC CTD data file may have variables P, Brunt Vaisala frequency (EPICKEY variable code 90), Square of Brunt Vaisala frequency (EPICKEY variable code 91). The command to plot x=Square of Brunt Vaisala frequency and y=Pressure is “`evar,E91,P`”.

The **evar** command can be given without arguments to use default variable selection. For time series data, **evar** will yield a plot with x=date/time and y=the first variable following date/time on the data file. For CTD data, **evar** will yield a plot with x=variable in column 2 and y=variable in column 1 (usually pressure).

Valid *qualifiers* are:

:[no]offset For time series data. Controls whether *PPLUS* offsets the time word so that data points are plotted in the center of each time interval. The default is **:offset,** which is appropriate for most EPIC time series. (EPIC time words represent the start of the time interval in most cases, such as average data.) Use **:nooffset** to force *PPLUS* to plot data points at the start of each time interval (e.g., this would be appropriate for subsampled data). Default is **:offset**.

:[no]time For time series data. Controls whether *PPLUS* reads the time word from the time series data file. The default is **:notime**, which means that the data is evenly spaced in time, making it unnecessary to read the time words. Use **:time** to make *PPLUS* read the time word for data which is unevenly spaced in time. Default is **:notime** (unless dt is negative, in which case the default is **:time**).

:[no]next **:next** indicates that the next variable is to be read from the same data file. When **:next** is used, no new data file name will be read from the EPIC file. The variables indicated by the **evar** command will be read from the last data file. This option permits overplotting several variables from the same data file, and can be used with the commands described in the Advanced Commands chapter to produce a plot with multiple axes. When **:next** is used, both x and y variables must be specified with the **evar** command. Default is **:nonext**.

The above *qualifiers* will also work with the **vars** command when EPIC data is being read.

exit

Causes all output buffers to be flushed and exits the program.

fill *:qualifier color*

Creates an area filled polygon using the coordinates specified by a line.

color negative values correspond to pen colors, positive values to the palette entry value, a value of zero corresponds to white.

Valid *qualifiers* are:

:[no]wait Controls whether *PPLUS* pauses after plot completion. Pause is signaled by a tone and terminated by typing a character. If an <ESC> is typed *PPLUS* will return from the current command level to the lowest command level. Default = **:wait**.

:[no]overlay Controls whether *PPLUS* overlays this plot on the preceding plot. The axes and their labels are not redrawn. Moveable labels (**labs** command) will redraw. The default is **:nooverlay** which causes this plot to be a new plot.

flush

Flushes the plot file buffer. After the **flush** command the pplus meta file will be complete.

format *:qualifier frmt*

Allows the input of a user supplied format for formatted sequential data files. Null entry is not allowed. The current format is in global symbol ppl$format.

frmt a format, default=(3F10.2)
free for free form
pplus for DSF file with a *PPLUS* header
dsf for DSF files
bibo for DSF files without a BIBO header

epic for EPIC time series data
unf for UNFORMATTED files.

Valid *qualifier* (for EPIC data only) is:

:[no]ctd Controls whether EPIC data is read as time series data or as CTD data. If the data is EPIC CTD data, then the **:ctd** switch must be used. Default is **:noctd**.

get *file_name*

Restores options to those in effect at the time "`save,file_name`" was called. *file_name* must be specified.

grid [**linear**]

If the argument **linear** is omitted (default), normal gridding is used (see the *cay* parameter on the **conset** command to change the default combination of Laplacian and cubic spline smoothing. Otherwise, if **linear** is included, gridding is done by linear interpolation with the following restrictions on the data:

- Data must be on a grid. The grid may have irregular spacing.
- There cannot be gaps in the middle of the grid. Every grid point in the middle of the grid must be specified. For example, if you are contouring with water depth on the y-axis, be sure the depth interval is the same for all your data.
- The grid may have ragged edges.

Must be issued before the **rd** command. Note that if the grid is coarser than the data, it is possible that some of the data will not be used in the gridding process. It is best to make the grid as fine as or finer than the data rather than coarser.

help *arg*

Give access to the VAX/VMS help files on topic *arg*.

hlabs *n, height*

Sets the height to *height* in inches of the *n*th moveable label. The height is reset to the default (specified by the **labset** command) by omitting the height value or clearing the labels with a **labs** command. (Also see **labs**, **rlabs**, **llabs**, **labset**.) See **axlsze** and **txlsze** for setting numeric label height.

hlp *arg*

Gives help on the *PPLUS* topic *arg* in the VAX/VMS environment.

if *expression* **then**

The first element of a block **if** statement; the other two elements are **else** and **endif**. **else** and **endif** are not valid in any other context.

expression *argument operator argument*
argument symbol name, number or a string enclosed by quotes
operator **.eq.**, **.ne.**, **.lt.**, **.gt.**, **.le.**, or **.ge.**

The symbol name can be undefined and its value is then "" (i.e., null string).

inc *sym*

Increments the value stored in the symbol *sym* by one. If *sym* does not exist it is created and given a value of one.

labs *:qualifier n, x, y, jst, label*

Defines the *n*th movable label for all plots. With X11 when a moveable label is requested without supplying the coordinates a single cross locator prompt will appear when the mouse cursor is in the graphics window. To select left justification, centering or right justification of the label with respect to the locator prompt click the left, middle or right mouse button, respectively. To select the start of a line with an arrow head click the left button while pressing the shift key and to select the start of a plain line click the middle button while pressing the shift key. If you have indicated that you want a line a locator prompt will appear again this time as a double cross, reposition the locator and click the mouse button that corresponds to the type of justification you want. Null entries are not allowed for n or label. A comment will be inserted into the echo.file file giving the coordinates when cross hairs are used. If n is omitted **labs** is reset and all moveable labels are cleared. Also see **labset** (character heights), **hlabs** (character heights for individual labels), **rlabs** (rotation of individual labels), **llabs** (moveable labels with a line or arrow).

n	label number (up to 25 allowed)
x	X position of label in user units (optional)
y	Y position of label in user units (must exist if X is present)
jst	justification of label. -1 left (default), 0 center, +1 right
label	any SYMBEL compatible string

:[no]user	determines units of x and y positions. Default is **:user**. If **:nouser**, units are inches from the origin. (see the **origin** command)

NOTE: Units specified by the **:user** *qualifier* are also used in the **llabs** command. If your terminal does not have cross hairs, you must specify X and Y.

labset *hlab1, hxlab, hylab, hlabs*

Sets character heights for labels. (also see **labs**, **rlabs**, **llabs)**

hlab1	main label, default=.16 inches
hxlab	x - label, default=.12 inches
hylab	y - label, default=.12 inches
hlabs	movable labels, default=.12 inches

levels *n*

Sets the number of auto contouring levels. For use with the **area** and **pixel** color area fill commands. Sets the levels the way the **%range** command sets axis min, max, and tic intervals.

n	Number of contour levels to be used when auto contouring is chosen.

lev *arg, arg, arg ...*

Sets the contour levels, the contour line type, the contour line label characteristics and lets the user edit (insert/delete) levels. Any duplicate levels will be deleted, however, each **lev** command edits the existing levels and unless requested the levels are not cleared. Before you can use **dash**, **dark**, **del**, **line,** or

pen the levels must first be defined. Maximum number of levels is 500. At the end of the description of this command there are some examples of the **lev** command.

arg **()** clear levels, number of automatic levels to 10.

arg **(***min,max,inc,idig***)** specifies the contour levels and label type
min starting value for levels creation
max ending value for levels creation (if omitted only the starting level will be created)
inc increment used to create levels. (if omitted only the starting and ending levels will be created, if 0 the starting and ending levels are deleted)
idig 0 through 9. Number of digits after the decimal point in the label
-1, contour label plotted as an integer
-3, no contour label will be drawn

arg *type***(***min,max,inc,ipen***)** sets the contour lines specified to *type*
type **dash** sets the line type to dash
dark sets the line type to dark (heavy)
del deletes the indicated levels.
line sets the line type to line (normal)
pen sets the pen used for a contour line to *ipen*. *ipen*=0 to use default pen.

For example, “`lev,(),(9,20,1,-1),dash(8,20,2)`“ will clear the previous levels and create contours at every integral value from 9 to 20 with the labels drawn as integers, all even valued contours from 8 to 20 will be drawn with dashed lines. (NOTE: levels are defined before they can be modified with **dash**). Here is another example which sets pen numbers to allow contouring different levels in different colors:

```
lev (),(28.,36.4,.1,1),dark(28,36,1)
lev pen(28.,34.0,.1,2),pen(34.1,34.5,.1,4)
lev pen(34.6,35.0,.1,3),pen(35.1,35.2,.1,5)
lev pen(35.3,35.4,.1,6),pen(35.5,35.6,.1,7)
lev pen(35.7,40.0,.1,1)
```

Here is another example of the **lev** command which shows how each successive argument modifies the preceeding **lev** command. The *PPLUS* global symbols “blue”, “cyan”, etc., have been set to the pen numbers for blue, cyan, etc., on the desifed plotter.

```
lev (),(28.,36.4,.1,1),dark(28,36,1)
lev pen(28.,34.0,.1,’blue’),pen(34.1,34.5,.1,’cyan’)
lev pen(34.6,35.0,.1,’green’),pen(35.1,35.2,.1,’yellow’)
lev pen(35.3,35.4,.1,’red’),pen(35.5,35.6,.1,’magenta’)
lev pen(35.7,40.0,.1,’black’)
```

limits *value, comparison, flag*

This command sets the testing value and type of test for bad data points. X, y and z are checked and the point will not be plotted if the test is true.

value	test value for the test
comparison	**xle** test for x .le. *value*, default off, 0.0 **xeq** test for x .eq. *value*, default off, 0.0 **xge** test for x .ge. *value*, default on, 1.E35 **yle** test for y .le. *value*, default off, 0.0 **yeq** test for y .eq. *value*, default off, 0.0 **yge** test for y .ge. *value*, default on, 1.E35 **zle** test for z .le. *value*, default off, 0.0 **zeq** test for z .eq. *value,* default off, 0.0 **zge** test for z .ge. *value*, default on, 1.E35
flag	**off** the test is disabled, otherwise the test is enabled.

If you are reading data to be contoured with ZGRID, the limits are checked only after interpolation. If you are using “`grid, linear`”, limits are checked before and after interpolation.

line *n, mark, type, xoff, yoff , dn1, up1, dn2, up2*

Sets the characteristics for each of the X-Y plot lines.

n	line number
mark	data mark (see list at end of manual, e.g. 1 for x, 3 for +)
type	type of line 0 - line connecting points and no mark at each point 1 - mark data points 2 - mark end points only 3 - only mark (no line) 4 - dashes 5 - dashes with mark at end points
xoff	X offset, default=0.0
yoff	Y offset, default=0.0
dn,up	dash characteristics in inches.

Default *type*=0 for *n*=1, *type*=4 otherwise.

linfit *n, ximin, ximax, xomin, xomax*

A linear least squares fit is performed on the data in line n and the resulting fitting line is placed in the next available line buffer. For example, “`rd,data.fil`” followed by “`linfit,1`” will place the fitting line from the regression of line 1 into buffer 2. You must do **linfit** on line 1 before plotting it, not after plotting it.

n	line number (no default)
ximin	min x value for the regression domain
ximax	max x value for the regression domain
xomin	min x value for the fitting line (default=*ximin*)
xomax	max x value for the fitting line (default=*ximax*)

Ximin and *ximax* default to the minimum and the maximum of the data. *Xomin* and *xomax* default to *ximin* and *ximax*, respectively. An alternate form for the command may be used when **taxis** is **on** and **time** has been used. It is:

linfit,*n,timin,timax,tomin,tomax*

Where the arguments are the beginning and ending times in Woods Hole format Wyymmddhhmm or Wyyyymmddhhmm, i.e., W198101121800 is 12-JAN-1981 18:00. The arguments have the same meanings and defaults as above.

The following global symbols are defined by **linfit:**

ppl$lf_r2	regression coefficient squared
ppl$lf_a	constant for fit (y = a + b*x)
ppl$lf_a_stdev	standard error of A
ppl$lf_b	constant for fit
ppl$lf_b_stdev	standard error of B
ppl$lf_var	total variance
ppl$lf_res_var	residual variance after fit

list *imin, imax, jmin, jmax, vcomp, arg*

List on the terminal the appropriate information. Null entry is not allowed if *arg* is not **data**. *imin, imax, jmin, jmax* only valid if *arg*=**data**. Defaults are to print the total plot buffer.

imin	min I for CONTOUR , start pt for X-Y
imax	max I for CONTOUR , stop pt for X-Y
jmin	min J for CONTOUR , start line for X-Y
jmax	max J for CONTOUR , stop line for X-Y
vcomp	vector component to be listed (**vector** command)
arg	**levels** contour levels and weights
	conset contour information
	data data currently in buffer
	datpt contour data location before gridding
	labels prints the labels at the terminal
	labset prints the **labset** command parameters
	lines current **line** and **pen** command values
	limits the current values set/reset by the limits command
	plot gives plot information and plot file name
	read sequential read information
	stats min and max plus sizes of last read
	taxis T-axis attributes
	tics Tic sizes and options
	transxy X and Y transform values
	vector Vector plotting attributes (**vector** command)
	xaxis X-axis attributes
	yaxis Y-axis attributes

listsym

Lists the symbols currently defined.

llabs *n, x, y, type*

Defines the starting position in user units for a line associated with the moveable labels. The end of the line is determined from the **labs** command. This command has no effect if the label is to be positioned with the cross-hairs. If the command is issued without coordinates the *type* is set to none. Fancy has an arrow head at the starting position. (also see **labs**, **rlabs**, **hlabs**, **labset**)

n label number less than 11
x X position of line in user units
y Y position of line in user units
type line type. 0 no line, 1 normal line, 2 fancy line

NOTE: Units of x and y positions are determined by the **:user** *qualifier* in the **labs** command.

loadcmap *cmapfile*

Loads a color map for use with the **area**, **pixel**, **colorbar**, and **fill** area fill commands. There are color maps in the PPL$CMAP directory. You can also make your own color map.

cmapfile Name of the file containing a color map. The file will be read in free format. It should contain R,G,B triplets (3 numbers per line) with values from 0 to 255.

markh *n, size*

Sets the mark size used for plotting line number *n*. The mark size for line 1 is used for the marks in the **datpt** command (contouring).

n line number (no default)
size size of mark in inches (default= 0.08)

metric

Sets the internal conversion factors to millimeters. Default condition is inches.

mercat { **on** | **off** }

Turns Mercator projection **on** and **off**. Default = **off**. Supported commands with “`mercat,on`” are **plot**, **contour**, **area**, **pixel**, **vector**, and **labs**. Unsupported commands with “`mercat,on`” are **plotuv**, **plotv**, and **llabs**.

multplt *nx, ny*

This command allows the user to plot several plots together. The individual plots are arranged in rows and columns. The X axis length of each plot in the same column and the Y axis length of each plot in the same row are identical. The axis lengths are specified in rows and columns. The spacings between the rows and columns are also user controlled. If the spacing is zero the plots are placed together without axis labels if appropriate. There are prompts for all additional information needed. NOTE: The advanced commands and the **:overlay** switch are not compatible with this command.

nx number of columns
ny number of rows

The prompts will be:

```
ENTER XLEN FOR COLS 1,2,...,NX
multplt>
ENTER YLEN FOR ROWS 1,2,...,NY
multplt>
ENTER PLOT SPACINGS LEFT BNDRY TO COL1, COL1 TO COL2,ETC...
multplt>
```

```
ROW1 TO ROW2,...,ROW NY TO BOTTOM
multplt>
```

Axis length and the origin are reset after plotting is finished. Limited to 10 columns by 10 rows.

nlines

Resets the input buffer so that the next data line read will be line 1. The input buffer is normally reset when a plot is made.

origin *xorg, yorg*

Sets the distance the lower left hand corner of the plotting area is from the lower left corner of the box. The values of *xorg* and *yorg* are placed in the global symbols ppl$xorg and ppl$yorg.

xorg x-distance (in), default=1.4
yorg y-distance (in), default=1.2

pen *n, ipen*

Sets the pen to be used for line *n. ipen* should be in a range appropriate to the limitations of the plotting device. On the VERSATEC, pen 2 is thicker than pen 1, pen 5 is thicker than pen 4, etc. The pen selected for line 1 will be used to draw the contour lines. (also see **lev**)

n line number. If *n*=0 sets the pen used to plot the axes and labels.
ipen pen number. Default=1

peters { **on** | **off** }

Turns Peters projection **on** and **off**. Default = **off**. (See comments for **polar** and **mercat** commands.)

pixel *:qualifier vcomp, angle, label*

Does an area-filled contour plot of data in buffer. **pixel** puts an area filled rectangle at each grid point, centered on the grid point. **area** makes get a smoother area filled plot, but **pixel** is faster, both interactively and on the hard copy devices. **pixel** and **area** are like the **contour** command except **contour** draws contour lines instead of producing an area filled plot. Use **colorbar** to put a colorbar on the graph. Other commands related to area filled contour plots are **cbaxis**, **cbcsze**, **cbfor**, **cblabp**, **cblint**, **loadcmap**, **lev** and **levels**.

The label will replace that in the current main label buffer. label is optional. If either axis is log that index must be equally spaced in log-space. (i.e. 10**(xmin+dx)) **pixel** does not take the log of the coordinate. The label cannot begin with a numeric character, i.e., 95W. You can plot a number by specifying a font, e.g., @SR100 meters.

vcomp indicates which vector component to contour. Default is 1. *Vcomp* is to be used when a vector field has been read in. See the **vecset** and **vector** commands.
angle The angle used to rotate the contour plot (counter-clockwise) relative to the X and Y axes (degrees).

Valid *qualifiers* are:

:[no]wait Controls whether *PPLUS* pauses after plot completion. Pause is signaled by a tone and terminated by typing a character. If an <ESC> is typed *PPLUS* will return from the current command level to the lowest command level. Default = **:wait**.

:[no]overlay Controls whether *PPLUS* overlays this plot on the preceding plot. The axes and their labels are not redrawn. Moveable labels (**labs** command) will redraw. The default is **:nooverlay** which causes this plot to be a new plot.

plot *:qualifiers label*

Does an X-Y plot of data in the plot buffer (all lines). The plot label “*label*” is optional. The plot label can be blanked with the **title** command. If either x-axis or y-axis is log, **plot** will take the logarithm of the appropriate coordinate as it is plotted. This will not affect the data buffer.

Valid *qualifiers* are:

:[no]wait Controls whether *PPLUS* pauses after plot completion. Pause is signaled by a tone and terminated by typing a character. If an <ESC> is typed *PPLUS* will return from the current command level to the lowest command level. Default = **:wait**.

:[no]overlay Controls whether *PPLUS* overlays this plot on the preceding plot. The axes and their labels are not redrawn. Moveable labels (**labs** command) will redraw. The default is **:nooverlay** which causes this plot to be a new plot.

plotv *:qualifiers vang, inc, label*

Does a stick plot for U,V pairs stored in X,Y respectively. May be used with or without **taxis** option **on**.

vang	rotation angle of vectors, default=0.0
inc	plots every inc vector (subsamples)
label	plot label

Valid *qualifiers* are:

:[no]wait Controls whether *PPLUS* pauses after plot completion. Pause is signaled by a tone and terminated by typing a character. If an <ESC> is typed *PPLUS* will return from the current command level to the lowest command level. Default = **:wait**.

:[no]overlay Controls whether *PPLUS* overlays this plot on the preceding plot. The axes and their labels are not redrawn. Moveable labels (**labs** command) will redraw. The default is **:nooverlay** which causes this plot to be a new plot.

plotuv *:qualifiers vang, inc, label*

Similar to **plotv** except U and V are in alternate pairs, where X1= count, Y1= U component, X2= count, Y2= V component, etc.

Valid *qualifiers* are:

:[no]wait Controls whether *PPLUS* pauses after plot completion. Pause is signaled by a tone and terminated by typing a character. If an <ESC> is typed *PPLUS* will return from the current command level to the lowest command level. Default = **:wait**.

:[no]overlay Controls whether *PPLUS* overlays this plot on the preceding plot. The axes and their labels are not redrawn. Moveable labels (**labs** command) will redraw. The default is **:nooverlay** which causes this plot to be a new plot.

pltnme *fname*

Specifies the file name to be used for plots. File name is available in the global symbol ppl$pltnme. Complete file name will be *fname*XXX or *fname*XXX.*fname_ext*, if *fname_ext* is non-blank, and XXX is a sequence number generated by *PPLUS*.

fname	the file name (default = ppl.meta)

pltnmext *fname_ext*

Specifies the file extension to be used for plots. File name extension is available in the global symbol ppl$pltnmext. NOTE: *fname_ext* should not include the “.” dot character.

fname_ext	the file name extension (default = *empty string*)

pltype *icode*

Sets plotting medium. Null entry is not allowed. The binary file is converted into device specific code using a post processor. The plot file name can be specified using the **pltnme** command.

icode	device code for plotting -2 = HP and TEK -1 = HP 0 = Binary file 1 = TEK 2 = TEK and Binary file 3 = X11 4 = X11 and Binary file, default=1

polar *longll, latll, longur, latur,* { **on** | **off** }

Turns Polar Stereographic projection **on** and **off** and sets the projection limits. Supported commands with **polar** are **plot**, **contour**, **area**, **pixel**, **vector**, and **labs**. Unsupported commands are **plotuv**, **plotv**, and **llabs**.

longll	Longitude of lower left corner. (positive west)
latll	Latitude of lower left corner. (positive north)
longur	Longitude of upper right corner.
latur	Latitude of upper right corner.

quit

Causes all output buffers to be flushed and exits the program.

rd *:qualifier nx, ny, type, nsets, file_name*

Read formatted or unformatted data from a sequential file according to **format** and **vars** or **evars**. The input file name is available in the global symbol ppl$input_file. The data can be (x,y) pairs for an x-y plot, (x,y,z) triplets for a contour plot or, if the data is pre-gridded, simply z-values for a contour plot. If the (x,y) data is being read (no contouring), then the only arguments used are *nx*, *nsets* and *file_name*.

If the data is to be contoured, *nx* and *ny* define the number of points in the grid on which data will be placed before plotting.

When (x,y,z) triplets are being read, this grid can be coarser or finer than the input data grid. Thus, when reading triplets, *nx*=50, *ny*=21 indicates that the number of grid points to be used for contouring will be 50 x 21 (the input data need not be on this grid). The limits of the grid in x and in y are specified by *xmin, xmax, ymin, ymax*. The **rd** command prompts for these and they are described below. The data is gridded according to the **rd** command specifications before being placed in the *PPLUS* data buffer.

When the input data are z-values to be contoured, the input grid and the plotting grid must be identical and are described by *nx* and *ny*.

Maximum number of points for a single read is 100,000 pairs, 200,000 grid points or 50,000 triplets. Default number of points read is the remaining buffer space. *File_name* may be omitted if previously defined. Null entries are not allowed.

nx no. of columns on the plotting grid for contouring or no. of points to read if not contouring. See *ny* for explanation.

ny no. of rows if data is on a grid for contouring. Omitted otherwise.
The meaning of *nx* and *ny* change depending on whether you're reading data for contouring or not. If you're reading contour data *nx* is the number of columns and *ny* is the number of rows.
If the data is not contour data *nx* is the number of points to be read and *ny* is not required. The default for *nx* is the space remaining in the buffer. Reading will stop automatically at the EOF without any error.

type method by which grid data is to be read (contour data only)
0 by rows (1st subscript varies fastest)
1 by columns (2nd subscript varies fastest)

nsets number of data sets to be read (on same file).

file_name file name. (VMS default device is SY:.) If the file name is explicitly given the file will be read after rewinding the file. If the file name is not given no rewind takes place.
If the data is EPIC, the file name given with the **rd** command is the name of the EPIC/pointer file for the data file. Otherwise, the file name is the name of the data file itself

Valid *qualifier* (use only with **vector**, **vecset**, **veckey** commands):

:[no]vector **:vector** reads the second component of a vector field to be gridded using the old *xmin,xmax,ymin,ymax*. This is done after the first vector component has been read in the usual fashion. (If the vector data is (u,v) pairs, **rd** in the usual fashion reads the u-component. Then "`rd:vector nx,ny`" reads the v-component. Do not enter *npoints* or *xmin,xmax,ymin,ymax* when using **rd:vector**.) See the **vector** command. The default is **:novector**.

If you are reading triplets *PPLUS* prompts for total number of points to be read in with "`rd>`". If you are reading triplets or grid data *PPLUS* will also prompt for *xmin,xmax,ymin,ymax*, which are the limits of the plotting grid.

reset

Uses the logical PPL$RESET as the input file to the **get** command.

return

Return from current command level to the previous command level. If executed at the top level *PPLUS* will exit.

rlabs *n, ang*

Specifies the angle to rotate the moveable labels. (The labels defined by the **labs** command.)

n	number of the label (no default)
ang	angle in degrees. Default = 0.0

rotate { **on** | **off** }

Rotates the plot 90 degrees on the screen and plotter. Default = **off**

rwd *file_name*

Rewinds the current data file. *File_name* may be omitted if previously defined. Files are also rewound by explicitly including the file name in the **skp** and **rd** commands. Rewinds the EPIC pointer file. The input file name is available in the global symbol ppl$input_file.

If the data is EPIC, the file name given with the **rwd** command is the name of the EPIC/pointer file for time series data. Otherwise, the file name is the name of the data file itself.

save *file_name*

Saves the options currently in effect on file *file_name* in a binary format. *File_name* must be specified.

set *sym arg*

Creates/modifies the symbol *sym* and sets it to *arg*. The argument *arg* can be either a legal character string, a simple arithmetic expression, or a special function. A simple arithmetic expression is of the form *num1 op num2*, where *op* is +, -, * or / (addition, subtraction, multiplication or division) and *num1* and *num2* are numbers. The numeric values must be separated from the operator op by spaces. The string will be used exactly as it appears if enclosed by double quotes (“). For example:

```
SET XPOS 4.4 + 2 results in XPOS = 6.200E00
SET A_LABEL “4.4 + 2” results in A_LABEL = 4.4 + 2
```

The special functions manipulate and reformat character strings. They are:

$edit(symbol,argument)
$extract(start,length,symbol)
$integer(symbol)
$length(symbol)
$locate(substring,symbol)
$element(position,delimiter,symbol)

The general fromat is **set** sym $function(arg1, arg2, ...). These functions are described in the SPECIAL FUNCTIONS section.

show *symbol*

Prints the current value of “*symbol*”.

size *width, height*

Sets total plotting size in inches of the plotting region. Null entries are not allowed. The width and height should be about 2 and 1.5 inches greater than the respective axis lengths. The displacement specified by **origin** must be considered when values for **size** and **axlen** are being chosen. The maximum allowed size for Versatec plots (to keep the plot on a single page) is 8 by 10.5. The values of width and height are placed in the global symbols ppl$width and ppl$height.

width	plotting area total width (default = 7.5)
height	plotting area total height (default = 5.625)

skp *n, file_name*

Skip *n* sequential or unformatted records. *File_name* may be omitted if previously defined. If the file name is explicitly given the records will be skipped after rewinding the file. If the file name is not given no rewind takes place. The input file name is available in the global symbol ppl$input_file.

If the data is EPIC, the file name given with the **skp** command is the name of the EPIC/pointer file for time series data. Otherwise, the file name is the name of the data file itself.

smooth *n*

Does *n* laplacian smoothings on contour type data. Null entry is not allowed.

spawn

Creates a VAX/VMS sub-process and passes control to this process. When finished with the spawned process type LOGOUT to return to *PPLUS*.

sqfill { **on** | **off** }

Sets **area** fill routine to square fill or triangle fill. Default is square fill (**on**).

taxis *:qualifier dt, arg*

Sets the time axis characteristics. The axis length is specified with **axlen**for this style axis. When **taxis** is turned on and BIBO or EPIC formatted data is read, the time series are automatically adjusted properly relative to *tmin*. NOTE: *dt* and *tstart* (set with the **time** command) are needed only when BIBO or EPIC data is not being used.

dt	sampling rate in minutes (default=1440, i.e., daily)
arg	turns **taxis** option on and off (default=off)

Valid *qualifiers* are:

:[no]yaxis	if yaxis draw a vertical time axis in place of the yaxis. (**:noyaxis**)

teknme *[fname]*

Stores the Tektronix plot in file *fname* if specified. Terminal must have NOWRAP to dump the plot back to the screen with the TYPE command. The current Tektronix plot file name is available in global symbol ppl$teknme

tics *smx, lgx, smy, lgy, ix, iy*

Sets the sizes in inches of the small and large tics on the X and Y axis. The tic style may also be set for both axis.

smx small X axis tic size, default=0.125
lgx large X axis tic size, default=0.25
smy small Y axis tic size, default=0.125
lgy large Y axis tic size, default=0.25
ix 1 X tics on the inside
0 X tics on both sides
-1 X tics on the outside (default)
iy 1 Y tics on the inside
0 Y tics on both sides
-1 Y tics on the outside (default)

time *tmin, tmax, tstart*

Specifies time axis limits and starting time of time series data. See **txtype** command for restrictions. (Default is auto-scaling of the time axis for BIBO and EPIC formatted data)

tmin Start date time of time axis (WHOI format = Wyymmddhhmm or Wyyyymmddhhmm)
tmax End date time of time axis
tstart Start time of time series data (optional, see following)

Note: If you read time as a sequence number and specify *dt* (set with the **taxis** command) and *tstart*, then the *tstart* time/date must correspond to a sequence number of 1. *Tmin* and *dt* (see **taxis** command) must be specified before *tstart*. *Tstart* must be re-entered whenever *dt* is changed.

title *hlab, label*

Sets the main plot title to “*label*” without generating a plot. If “*label*” is omitted the main plot title is cleared. Optionally the size of the title can also be specified.

hlab the height of the title in inches. (default = .16 inches)

tktype *type*

Sets the type of TEK terminal. Null entry is not allowed. Valid values are: 4010, 4014, 4107, 4115, 4051, 4052 and 4662.

type model no. of TEK terminal, default=4010

transxy *n, xfact, xoff, yfact, yoff*

Lets you define a linear transformation for the X and Y variables in each line, i.e., xt(i) = *xfact**x(i) + *xoff*. **transxy** does not affect the data. The translation is only applied as the data is plotted.

n line number (no default)
xfact multiplicative factor for X (default=1.0)
xoff offset for X (default=0.0)
yfact multiplicative factor for Y (default=1.0)
yoff offset for Y (default=0.0)

The transformation factors are available in the global symbols ppl$xfact(*n*), ppl$xoff(*n*), ppl$yfact(*n*), and ppl$yoff(*n*), where *n* is the line number. Initially only the first 10 lines will have these symbols defined.

If the value being scaled is time and **taxis** is **on**, *xoff* or *yoff* is in units of *dt*. Unless *dt* is changed with the **taxis** command, it will have the default value of 1 day.

txlabp *n*

Specifies time axis label position (-1 for below plot, 0 for no label, or +1 for above plot).

txlint *low_int, hi_int*

Specifies which time axis tics will be labeled.

low_int labeling interval for lowest level of tics (e.g. 0 for no month labels, 1 for every month, 2 for every other month on mon/yr axis)
hi_int labeling interval for highest level of tics (e.g. 0 for no year labels, 1 for every year, 2 for every other year on mon/yr axis)

txlsze *ht*

Specifies height of time axis labels (inches).

txnmtc *n*

Specifies number of small tics between large tics on time axis. If *n* is -1 the major divisions are denoted by large tics and the minor divisions by small tics, otherwise they are denoted by thick tics and large tics, respectively.

txtype *type, style*

Specifies type and style of time series axis. Selection of an axis type with the **txtype** command places limitations on the arguments *tmin,tmax* of the **time** command. If *type* is **days**, then *tmin,tmax* must end in 00, the start of an hour. If *type* is **mon**, then *tmin,tmax* must end in 0000, the start of a day. If *type* is **yr**, then *tmin,tmax* must end in 010000, the start of a month.

type **hours**
style **min** (hour, minute on 2 lines) (default)
hrmin (on 1 line)
days
style **hr** (hour,day on 2 lines) (default)
hrday (on 1 line)
mon
style **day** (day,mon on 2 lines) (default)
daymon (day,mon on 1 line)
yr (default)

style **mon1** (1-char month)
mon3 (3-char month) (default)
monyr (month,yr on 1 line)

usr *local_command*

The **usr** command is implemented by linking a locally created subroutine named pplusr with the *PPLUS* program. The calling arguments are:

```
subroutine pplusr(command,ier,msg)
character command*255,msg*80
integer ier

... etc ...

return
end
```

The subroutine pplusr is expected to exit normally with ier=0. If ier is not 0 then msg is displayed and placed in the echo file. The format for *local_command* is user determined. The pplusr subroutine communicates with *PPLUS* via the subroutines pplldc (load contour data), pplldx (load x-y pairs) and putsym (create a *PPLUS* global symbol). See local documentation for the format of the **usr** command.

vars *ngrp, a1, a2, a3 , .. ai*

Defines the location of variables within a record of a sequential data file. If only a single variable is specified and it is either X or Y the other is automatically filled with the data point number. If only Z (gridded data) is given the program expects data to be grid points in one of two formats, by rows or by columns. If X, Y, and Z (triplets) are given the program uses ZGRID to put the data on a evenly spaced grid. See the chapters Getting Started and Data Formats for more information on VARS.

ngrp no. of groups per record
aj 1,2, or 3 The position of *Aj* in **vars** command indicates which variable is to be read as an x, y or z.
1 = X variable
2 = Y variable
3 = Z variable
i NVAR no. of variables per group. Default = "`vars,1,1,2`" (i.e. one group per record, first variable is X, second is Y). If left blank indicates a number not to be read, but a variable is present and expected by the **format**.

veckey *:qualifier x, y, ipos, format*

veckey sets where the scaling key for the vectors is plotted. See **vector** and **vecset** commands.

x x position of vector key
y y position of vector key (default is no key at all)
ipos relative position of key (not implemented)
format format to draw the numeric part of the key, default = (1pg10.3)

Valid *qualifiers* are:

:[no]user determines units of *x* and *y* positions. Default is **:user**. If **:nouser** units are inches from the origin. (see the **origin** command)

vecset *length, scale*

vecset sets the scaling for the vectors plotted. See the **vector** and **veckey** commands.

length length of standard vector in inches. this is also the length of the scale vector. Default is 0.5.

scale length of standard vector in user units. This is also the length of the scale vector is user units. Default is the twice the mean length of the vectors.

vector *:qualifier skipx, skipy, angle, label*

vector draws a field of vectors from two-component grids. The data for each component is read in just as scalar data is read in for contouring, with the **rd** command. The first vector component is read with the **rd** command, then the second component is read with the “`rd:vector`” command. See the **rd, veckey** and **vecset** commands.

skipx plot every *skipx* column (default is 1)
skipy plot every *skipy* row (default is 1)
angle angle to rotate the vectors (degrees) (default is 0)
label title of plot

Valid *qualifiers* are:

:[no]wait Controls whether *PPLUS* pauses after plot completion. Pause is signaled by a tone and terminated by typing a character. If an <ESC> is typed *PPLUS* will return from the current command level to the lowest command level. Default = **:wait**.

:[no]overlay Controls whether *PPLUS* overlays this plot on the preceding plot. The axes and their labels are not redrawn. Moveable labels (**labs** command) will redraw. The default is **:nooverlay**, which causes this plot to be a new plot.

view *:qualifiers zscale, ic, zmin, zmax, vcomp, labe*l

Does a 3 dimensional surface plot. *Label* is optional.

zscale scale of the z data, default=(ymax - ymin) / (zmax - zmin)
ic 0 set Xscal = Yscale, =1 no effect. Default=0
zmin set the base of the surface plot to *zmin*. Default: use *zmin* from the data
zmax set the top of the surface plot to *zmax*. Default: use *zmax* from the data
vcomp Vector component to use for plotting (see the **vector** command). Default is 1.

Valid *qualifiers* are:

:[no]wait Controls whether *PPLUS* pauses after plot completion. Pause is signaled by a tone and terminated by typing a character. If an <ESC> is typed *PPLUS* will return from the current command level to the lowest command level. Default = **:wait.**

:[no]overlay Controls whether *PPLUS* overlays this plot on the preceding plot. The axes and their labels are not redrawn. Moveable labels (**labs** command) will redraw. The default is **:nooverlay**, which causes this plot to be a new plot.

Best results are normally obtained by using defaults. Using scales does not change the data buffer.

vpoint *x, y, z*

Sets the viewpoint coordinates for surface plotting. To create a surface plot use the **view** command. The viewpoint coordinates are available in the global symbols ppl$view_x, ppl$view_y and ppl$view_z. X, Y and Z form a right handed coordinate system with the Z axis up and Y axis into the page.

x x coordinate of viewpoint
y y coordinate of viewpoint
z z coordinate of viewpoint

while *expression* **then**

The first element of a **while** statement the other element is **endw**. **endw** is not valid in any other context.

expression *argument operator argument*
argument symbol name, number or a string enclosed by quotes
operator **.eq**., **.ne**., **.lt.**, **.gt.**, **.le.**, or **.ge.**

The symbol name can be undefined and its value is then “” (i.e., null string).

window { **on** | **off** }

Windows the data to within the axes. Default=**off**

write *vcomp, angle, file*

Writes a grid to a file as triplets. The format used is (4G15.5). The data in each record is the x-position, y-position, z-first component, z-second component. The grid coordinates will be rotated about the center of the grid by angle degrees counter-clockwise.

vcomp indicates which vector component to write to the file. Default is 0, write both components. *Vcomp* is to be used when a vector field has been read in.
angle The angle used to rotate the grid (counter-clockwise) relative to the X and Y axes (degrees) when writing. No default. Will use last angle specified.
file Name of file to create.

xaxis *xlo, xhi, xtic*

Sets the x-axis characteristics. If *typex*, from the **axtype** command, is not 1, then *xlo* and *xhi* must be the log of the minimum and maximum. (must be integral values). **xaxis** without arguments resets the auto scaling. Auto scaling does consider **limits** and does not consider “`window,on`”.

xlo axis minimum (beginning of axis)
xhi axis maximum (end of axis)
xtic distance between large tics

xfor *frmt*

Sets the format for the x axis label.

frmt 0 or (a format), default=0 (auto label) To create an integer numeric label the format must begin as “`(I`” or “`(i`”. A latitude or longitude axis can be created by specifying `''LAT'')`, `''LON'')`, `''LONE'')` or `''LONW'')` in the format. Two single quotes are required because *PPLUS* symbol substitution will occur with 1 single quote. The hemisphere designation will be inserted. Longitude must be continuous across the dateline with west positive for `''LON''` or `''LONW''`, i.e., 135 is 135W and 190 is 170E. For `''LONE''` longitude is continuous across the dateline with east positive, i.e., 135 is 135E and 190 is 170W.

xgeom *width, height, xorg, yorg*

Sets the size and location of the X11 window for graphics output. For this command to have any effect it must be called before the graphics window is opened.

width	window width in pixels. (default=800)
height	window height in pixels. (default=640)
xorg	window x position (default=-10)
yorg	window y position (default=10)

xlab *label*

Enters the x-axis label. *Label* is ignored if **taxis** is **on**.

yaxis *ylo, yhi, ytic*

Same as **xaxis**.

yfor *frmt*

Same as **xfor**.

ylab *label*

Enters the y-axis label.

13 Advanced Commands

This chapter describes *PPLUS* primitive plot commands. With these commands, the user can make a plot with several x- or y-axes. The location of each axis can be specified. To distinguish them from the standard *PPLUS* commands, these commands all begin with “**%**”.

These **%** commands can be entered only from a *PPLUS* command file, and can not be entered interactively from the keyboard. Each command is implemented as it is read from the command file. Specifically, when the **%xaxis** command is read from a command file, an x-axis is immediately drawn on the graph. By contrast, the standard *PPLUS* XAXIS command simply sets x-axis parameters and the x-axis is not drawn on the graph until a plotting command such as PLOT is issued. The **%** commands give the user great control over the graphics display, but must be used carefully. No *PPLUS* error messages are issued for illegal **%** commands. The **%** commands can not be used with the **multplt** command. See the notes with each command description and the example at the end of this chapter. Command descriptions follow. (Recall the VAX/VMS qualifier escape character is “/” and not the UNIX character “:”.)

%opnplt *:qualifier*

Opens the plot by putting the terminal into and out of graphics mode and setting **:quiet**.

Valid *qualifiers* are:

:[no]overlay Controls whether *PPLUS* overlays the next plot on the preceding plot. The default is **:overlay**, which causes the next plot to be overlaid without erasing the last plot.

%clsplt *:qualifiers*

Closes the plot by putting the terminal out of graphics mode and restoring **:quiet** or **:noquiet**, whichever was in effect when the **%opnplt** command was issued.

Valid *qualifiers* are:

:[no]wait Controls whether *PPLUS* pauses after plot completion. Pause is signaled by a tone and terminated by typing a character. If an <ESC> is typed *PPLUS* will return from the current command level to the lowest command level. Default = **:wait**.

%pltlin *n*

Plots the *n*-th data line. Each **rd** command increments the data line count by 1. Use of the standard plotting commands (**plot**, **plotuv**, **plotv**, **contour**, **vector**, and **view**) resets the data line count. The **%pltlin** command does not reset the data line count. (**window** does work with this command.)

n Plot line n using current scale factors.

%label *:qualifier x, y, ipos, ang, chsiz, label*

Draws a label similar to a moveable label (**labs** command). There is no label number and the label is drawn as soon as the command is read from the command file. Any number of labels may be drawn.

x x position user or inches
y y position user or inches
ipos -1 left, 0 center, +1 right justify
ang Angle at which label is to be drawn. (0 degrees is at 3 o’clock and positive rotation is counter clockwise.)
chsize character size (inches)
label character string to draw

Valid *qualifiers* are:

:[no]user determines units of x and y positions. Default is **:user**. If **:nouser**, units are inches from the origin. (see the **origin** command)

%noaalogo *:qualifier x, y, size*

Draws a NOAA logo, without any text, at *x, y* and a diameter of *size* inches.

x x position user or inches
y y position user or inches

Valid *qualifiers* are:

:[no]user determines units of x and y positions. Default is **:user**. If **:nouser**, units are inches from the origin. (see the **origin** command)

%range *min, max, ntic*

Finds axis limits for use with the **%xaxis** and **%yaxis** commands given the data extrema of *min* and *max*. The axis limits and tic interval are returned in the *PPLUS* symbols ppl$range_low, ppl$range_high, and ppl$range_inc.

min minimum value of data to be ranged
max maximum value of data
ntic number of large increments

The *PPLUS* global symbols ppl$range_low, ppl$range_high and ppl$range_inc contain the new low, high and increment.

%xaxis *:qualifier xlow, xhigh, xtic, y* [*, nmstc*] [*, lint*] [*, xunit*] [*, ipos*] [*, csize*] [*, frmt*]

This command draws an x-axis and redefines scaling for the x-direction. The arguments xlow, xhigh, xtic and y should not be omitted. See the **%range** command to get default values for axis limits and increments. If you have used the command **%range**, then you can use the symbols ppl$range_low, ppl$range_high, ppl$range_inc for the values of *xlow*, *xhigh* and *xtic*.

xlow	min value of x user
xhigh	max value of x user
xtic	large tic increment user
y	y position user or inches
nmstc	number of small tics
lint	label interval (large tics)
xunit	divisor for axis label
ipos	-1 bottom, 0 none, +1 top of label
csize	character size inches
frmt	axis format char*20

Valid *qualifiers* are:

:[no]user determines units of x and y positions. Default is **:user**. If **:nouser**, units are inches from the origin. (see the **origin** command)

%yaxis *:qualifier ylow, yhigh, ytic, x* [*, nmstc*] [*, lint*] [*, yunit*] [*, ipos*] [*, csize*] [*, frmt*]

This command draws an y-axis and redefines scaling for the y direction. The arguments ylow, yhigh, ytic and x should not be omitted. See the **%range** command to get default values for axis limits and increments. If you have executed **%range**, then you can use the symbols ppl$range_low, ppl$range_high, ppl$range_inc for the values of ylow, yhigh and ytic. Normally the y-axis tics are labeled increasing upwards. If you want the tics to be labeled increasing downwards, use ytic negative where ylow is greater than yhigh.

ylow	min value of y user
yhigh	max value of y user
ytic	large tic increment user
x	x position user or inches
nmstc	number of small tics
lint	label interval (large tics)
yunit	divisor for axis label
ipos	-1 left, 0 none, +1 right of label
csize	character size inches
frmt	axis format char*20

Valid *qualifiers* are:

:[no]user determines units of x and y positions. Default is **:user**. If **:nouser**, units are inches from the origin. (see the **origin** command)

%vector *x, y, u, v*

Plots a single vector at user unit coordinates (*x*,*y*) and of length (*u*,*v*). The **vecset** command must be called prior to any **%vector** command.

x	x coordinate of origin
y	y coordinate of origin
u	x component of vector
v	y component of vector

Example

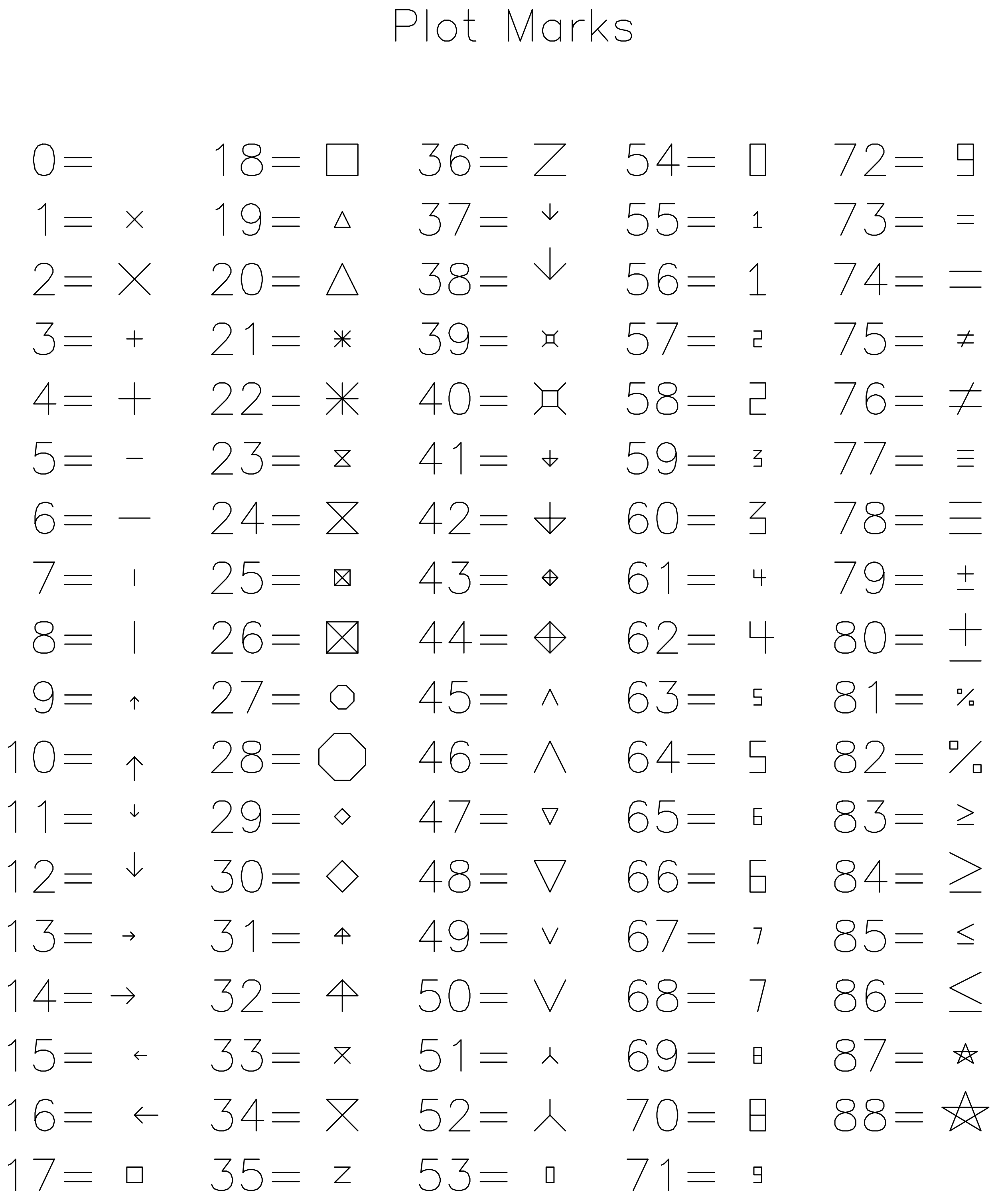
The following is a *PPLUS* command file which uses all the **%** routines described above. It can be found in the directory `ppl$examples` (`PPL$EXAMPLES: CTD4.PPC`), and can be executed in *PPLUS* to generate a plot.

```
c
c pplus command file to plot EPIC CTD data demonstrating
c multiple axis capability.
c
c It plots Pressure vs Temperature, Salinity,
c Sigma_t, Oxygen.
c
box,off
window,on
size,8,10.5
origin,,2.3
format:ctd,epic
axlint,1,1
c
pltnme,ctd4.plt
c
c First plot P vs T with T axis at top.
c Suppress bottom x axis.
c
evar,t,p
rd,ppl$examples:ctd4
%opnplt
%range:nouser ‘ppl$ymin(1)’,’ppl$ymax(1)’,5
yfor,(i7)
yaxis,’ppl$range_high’,’ppl$range_low’,’ppl$range_inc’
title
axlabp,1
axset,,0
plot
c
c Plot P vs Salinity with S axis at top above T axis.
c
evar:next sal,p
rd
set ypos ‘ppl$ylen’ + .7
%range:nouser ‘ppl$xmin(1)’,’ppl$xmax(1)’,4
```

```
%xaxis:nouser,’ppl$range_low’,’ppl$range_high’,-
‘ppl$range_inc’,’ypos’,,,,+1
%pltlin,1
c
c Plot P vs Sigma_t with S_t axis at bottom
c
evar:next sig,p
rd
set ypos 0.
%range:nouser ‘ppl$xmin(2)’,’ppl$xmax(2)’,4
%xaxis:nouser,’ppl$range_low’,’ppl$range_high’,-
‘ppl$range_inc’,’ypos’,,,,-1
%pltlin 2
c
c Plot P vs Oxygen with O axis at bottom below S_t axis.
c
evar:next ox,p
rd
set ypos ‘YPOS’ - .7
%range:nouser ‘ppl$xmin(3)’,’ppl$xmax(3)’,4
%xaxis:nouser,’ppl$range_low’,’ppl$range_high’,-
‘ppl$range_inc’,’ypos’,,,,-1
%pltlin 3
c
c Now use PPLUS EPIC symbols in moveable labels for
c graph titles
c
set ypost ‘ppl$ylen’ + 1.9
%label:nouser 0,’ypost’,-1,0.,.16,-
‘ppl$epic_latitude1’ ‘ppl$epic_longitude1’
set ypos ‘ypost’ + .3
%label:nouser 0,’ypos’,-1,0.,.16,-
‘ppl$epic_cast1’ ‘ppl$epic_date1’
%clsplt
```

A

Line Plot Marks


Plot Marks
0=
1=
2=
3=
4=
5=
6=
7=
8=
9=
10=
11=
12=
13=
14=
15=
16=
17=
18=
19=
20=
21=
22=
23=
24=
25=
26=
27=
28=
29=
30=
31=
32=
33=
34=
35=
36=
37=
38=
39=
40=
41=
42=
43=
44=
45=
46=
47=
48=
49=
50=
51=
52=
53=
54=
55=
56=
57=
58=
59=
60=
61=
62=
63=
64=
65=
66=
67=
68=
69=
70=
71=
72=
73=
74=
75=
76=
77=
78=
79=
80=
81=
82=
83=
84=
85=
86=
87=
88=

B

Character Fonts

SR — Simplex Roman

32	48 0	n.a.	80 P	96	112 p
33 !	49 1	65 A	81 Q	97 a	113 q
34 "	50 2	66 B	82 R	98 b	114 r
35 °	51 3	67 C	83 S	99 c	115 s
36 $	52 4	68 D	84 T	100 d	116 t
37	53 5	69 E	85 U	101 e	117 u
38 &	54 6	70 F	86 V	102 f	118 v
39 '	55 7	71 G	87 W	103 g	119 w
40 (	56 8	72 H	88 X	104 h	120 x
41)	57 9	73 I	89 Y	105 i	121 y
42 *	58 :	74 J	90 Z	106 j	122 z
43 +	59 ;	75 K	91	107 k	123
44 ,	60 ‘	76 L	n.a.	108 l	124
45 —	61 =	77 M	93	109 m	125
46 .	62 ’	78 N	n.a.	110 n	126
47 /	63 ?	79 O	n.a.	111 o	127

DR — Duplex Roman

32	48 0	n.a.	80 P	96	112 p
33 !	49 1	65 A	81 Q	97 a	113 q
34 "	50 2	66 B	82 R	98 b	114 r
35 °	51 3	67 C	83 S	99 c	115 s
36 $	52 4	68 D	84 T	100 d	116 t
37	53 5	69 E	85 U	101 e	117 u
38 &	54 6	70 F	86 V	102 f	118 v
39 '	55 7	71 G	87 W	103 g	119 w
40 (	56 8	72 H	88 X	104 h	120 x
41)	57 9	73 I	89 Y	105 i	121 y
42 *	58 :	74 J	90 Z	106 j	122 z
43 +	59 ;	75 K	91	107 k	123
44 ,	60 ‘	76 L	n.a.	108 l	124
45 —	61 =	77 M	93	109 m	125
46 .	62 ’	78 N	n.a.	110 n	126
47 /	63 ?	79 O	n.a.	111 o	127

TR – Triplex Roman

32	48 0	n.a.	80 P	96	112 p
33 !	49 1	65 A	81 Q	97 a	113 q
34 ″	50 2	66 B	82 R	98 b	114 r
35 °	51 3	67 C	83 S	99 c	115 s
36 $	52 4	68 D	84 T	100 d	116 t
37	53 5	69 E	85 U	101 e	117 u
38 &	54 6	70 F	86 V	102 f	118 v
39 ′	55 7	71 G	87 W	103 g	119 w
40 (	56 8	72 H	88 X	104 h	120 x
41)	57 9	73 I	89 Y	105 i	121 y
42 *	58 :	74 J	90 Z	106 j	122 z
43 +	59 ;	75 K	91	107 k	123
44 ,	60 ‘	76 L	n.a.	108 l	124
45 –	61 =	77 M	93	109 m	125
46 .	62 ’	78 N	n.a.	110 n	126
47 /	63 ?	79 O	n.a.	111 o	127

CR — Complex Roman

32	48 0	n.a.	80 P	96	112 p
33 !	49 1	65 A	81 Q	97 a	113 q
34 ″	50 2	66 B	82 R	98 b	114 r
35 ○	51 3	67 C	83 S	99 c	115 s
36 $	52 4	68 D	84 T	100 d	116 t
37	53 5	69 E	85 U	101 e	117 u
38 &	54 6	70 F	86 V	102 f	118 v
39 ′	55 7	71 G	87 W	103 g	119 w
40 (	56 8	72 H	88 X	104 h	120 x
41)	57 9	73 I	89 Y	105 i	121 y
42 *	58 :	74 J	90 Z	106 j	122 z
43 +	59 ;	75 K	91	107 k	123
44 ,	60 ‘	76 L	n.a.	108 l	124
45 —	61 =	77 M	93	109 m	125
46 .	62 ’	78 N	n.a.	110 n	126
47 /	63 ?	79 O	n.a.	111 o	127

AS — ASCII Simplex Roman

32	48 0	64 @	80 P	96 `	112 p
33 !	49 1	65 A	81 Q	97 a	113 q
34 "	50 2	66 B	82 R	98 b	114 r
35 #	51 3	67 C	83 S	99 c	115 s
36 $	52 4	68 D	84 T	100 d	116 t
37 %	53 5	69 E	85 U	101 e	117 u
38 &	54 6	70 F	86 V	102 f	118 v
39 '	55 7	71 G	87 W	103 g	119 w
40 (	56 8	72 H	88 X	104 h	120 x
41)	57 9	73 I	89 Y	105 i	121 y
42 *	58 :	74 J	90 Z	106 j	122 z
43 +	59 ;	75 K	91 [	107 k	123 {
44 ,	60 <	76 L	92 \	108 l	124 |
45 -	61 =	77 M	93]	109 m	125 }
46 .	62 >	78 N	94 ^	110 n	126 ~
47 /	63 ?	79 O	95 _	111 o	127

AC — ASCII Complex Roman

32	48 0	64 @	80 P	96 `	112 p
33 !	49 1	65 A	81 Q	97 a	113 q
34 "	50 2	66 B	82 R	98 b	114 r
35 #	51 3	67 C	83 S	99 c	115 s
36 $	52 4	68 D	84 T	100 d	116 t
37 %	53 5	69 E	85 U	101 e	117 u
38 &	54 6	70 F	86 V	102 f	118 v
39 '	55 7	71 G	87 W	103 g	119 w
40 (	56 8	72 H	88 X	104 h	120 x
41)	57 9	73 I	89 Y	105 i	121 y
42 *	58 :	74 J	90 Z	106 j	122 z
43 +	59 ;	75 K	91 [	107 k	123 {
44 ,	60 <	76 L	92 \	108 l	124 |
45 −	61 =	77 M	93]	109 m	125 }
46 .	62 >	78 N	94 ^	110 n	126 ~
47 /	63 ?	79 O	95 _	111 o	127

CS — Complex Script

32	48 0	n.a.	80 P	96	112 p
33 !	49 1	65 A	81 Q	97 a	113 q
34 ''	50 2	66 B	82 R	98 b	114 r
35 ○	51 3	67 C	83 S	99 c	115 s
36 $	52 4	68 D	84 T	100 d	116 t
37	53 5	69 E	85 U	101 e	117 u
38 &	54 6	70 F	86 V	102 f	118 v
39 '	55 7	71 G	87 W	103 g	119 w
40 (	56 8	72 H	88 X	104 h	120 x
41)	57 9	73 I	89 Y	105 i	121 y
42 *	58 :	74 J	90 Z	106 j	122 z
43 +	59 ;	75 K	91	107 k	123
44 ,	60 ‘	76 L	n.a.	108 l	124
45 —	61 =	77 M	93	109 m	125
46 .	62 ’	78 N	n.a.	110 n	126
47 /	63 ?	79 O	n.a.	111 o	127

TI — Triplex Italic

32	48 0	n.a.	80 P	96	112 p
33 !	49 1	65 A	81 Q	97 a	113 q
34 ″	50 2	66 B	82 R	98 b	114 r
35 °	51 3	67 C	83 S	99 c	115 s
36 $	52 4	68 D	84 T	100 d	116 t
37	53 5	69 E	85 U	101 e	117 u
38 &	54 6	70 F	86 V	102 f	118 v
39 ′	55 7	71 G	87 W	103 g	119 w
40 (	56 8	72 H	88 X	104 h	120 x
41)	57 9	73 I	89 Y	105 i	121 y
42 *	58 :	74 J	90 Z	106 j	122 z
43 +	59 ;	75 K	91	107 k	123
44 ,	60 ‘	76 L	n.a.	108 l	124
45 –	61 =	77 M	93	109 m	125
46 .	62 ’	78 N	n.a.	110 n	126
47 /	63 ?	79 O	n.a.	111 o	127

GE — Gothic English

32	48 0	n.a.	80 P	96	112 p
33 !	49 1	65 A	81 Q	97 a	113 q
34 "	50 2	66 B	82 R	98 b	114 r
35 °	51 3	67 C	83 S	99 c	115 s
36 $	52 4	68 D	84 T	100 d	116 t
37	53 5	69 E	85 U	101 e	117 u
38 &	54 6	70 F	86 V	102 f	118 v
39 '	55 7	71 G	87 W	103 g	119 w
40 (	56 8	72 H	88 X	104 h	120 x
41)	57 9	73 I	89 Y	105 i	121 y
42 *	58 :	74 J	90 Z	106 j	122 z
43 +	59 ;	75 K	91	107 k	123
44 ,	60 ‘	76 L	n.a.	108 l	124
45 —	61 =	77 M	93	109 m	125
46 .	62 ’	78 N	n.a.	110 n	126
47 /	63 ?	79 O	n.a.	111 o	127

IR – Indexical Complex Ro

32	48 0	n.a.	80 P	96	112 p
33 !	49 1	65 A	81 Q	97 a	113 q
34 "	50 2	66 B	82 R	98 b	114 r
35 °	51 3	67 C	83 S	99 c	115 s
36 $	52 4	68 D	84 T	100 d	116 t
37	53 5	69 E	85 U	101 e	117 u
38 &	54 6	70 F	86 V	102 f	118 v
39 '	55 7	71 G	87 W	103 g	119 w
40 (	56 8	72 H	88 X	104 h	120 x
41)	57 9	73 I	89 Y	105 i	121 y
42 *	58 :	74 J	90 Z	106 j	122 z
43 +	59 ;	75 K	91	107 k	123
44 ,	60 ʿ	76 L	n.a.	108 l	124
45 −	61 =	77 M	93	109 m	125
46 .	62 ʾ	78 N	n.a.	110 n	126
47 /	63 ?	79 O	n.a.	111 o	127

SS — Simplex Script

32	48	n.a.	80 P	96	112 p
33	49	65 A	81 Q	97 a	113 q
34	50	66 B	82 R	98 b	114 r
35	51	67 C	83 S	99 c	115 s
36	52	68 D	84 T	100 d	116 t
37	53	69 E	85 U	101 e	117 u
38	54	70 F	86 V	102 f	118 v
39	55	71 G	87 W	103 g	119 w
40	56	72 H	88 X	104 h	120 x
41	57	73 I	89 Y	105 i	121 y
42	58	74 J	90 Z	106 j	122 z
43	59	75 K	91	107 k	123
44	60	76 L	n.a.	108 l	124
45	61	77 M	93	109 m	125
46	62	78 N	n.a.	110 n	126
47	63	79 O	n.a.	111 o	127

CI — Complex Italic

32	48	n.a.	80 *P*	96	112 *p*
33	49	65 *A*	81 *Q*	97 *a*	113 *q*
34	50	66 *B*	82 *R*	98 *b*	114 *r*
35	51	67 *C*	83 *S*	99 *c*	115 *s*
36	52	68 *D*	84 *T*	100 *d*	116 *t*
37	53	69 *E*	85 *U*	101 *e*	117 *u*
38	54	70 *F*	86 *V*	102 *f*	118 *v*
39	55	71 *G*	87 *W*	103 *g*	119 *w*
40	56	72 *H*	88 *X*	104 *h*	120 *x*
41	57	73 *I*	89 *Y*	105 *i*	121 *y*
42	58	74 *J*	90 *Z*	106 *j*	122 *z*
43	59	75 *K*	91	107 *k*	123
44	60	76 *L*	n.a.	108 *l*	124
45	61	77 *M*	93	109 *m*	125
46	62	78 *N*	n.a.	110 *n*	126
47	63	79 *O*	n.a.	111 *o*	127

II — Indexical Complex It

32	48	n.a.	80 *P*	96	112 *p*
33	49	65 *A*	81 *Q*	97 *a*	113 *q*
34	50	66 *B*	82 *R*	98 *b*	114 *r*
35	51	67 *C*	83 *S*	99 *c*	115 *s*
36	52	68 *D*	84 *T*	100 *d*	116 *t*
37	53	69 *E*	85 *U*	101 *e*	117 *u*
38	54	70 *F*	86 *V*	102 *f*	118 *v*
39	55	71 *G*	87 *W*	103 *g*	119 *w*
40	56	72 *H*	88 *X*	104 *h*	120 *x*
41	57	73 *I*	89 *Y*	105 *i*	121 *y*
42	58	74 *J*	90 *Z*	106 *j*	122 *z*
43	59	75 *K*	91	107 *k*	123
44	60	76 *L*	n.a.	108 *l*	124
45	61	77 *M*	93	109 *m*	125
46	62	78 *N*	n.a.	110 *n*	126
47	63	79 *O*	n.a.	111 *o*	127

SG — Simplex Greek

32	48	n.a.	80 Π	96	112 π
33	49	65 Α	81 Θ	97 α	113 ϑ
34	50	66 Β	82 Ρ	98 β	114 ρ
35	51	67 Η	83 Σ	99 η	115 σ
36	52	68 Δ	84 Τ	100 δ	116 τ
37	53	69 Ε	85 ϒ	101 ε	117 υ
38	54	70 Φ	86	102 φ	118
39	55	71 Γ	87 Ω	103 γ	119 ω
40	56	72 Χ	88 Ξ	104 χ	120 ξ
41	57	73 Ι	89 Ψ	105 ι	121 ψ
42	58	74	90 Ζ	106	122 ζ
43	59	75 Κ	91	107 κ	123
44	60	76 Λ	n.a.	108 λ	124
45	61	77 Μ	93	109 μ	125
46	62	78 Ν	n.a.	110 ν	126
47	63	79 Ο	n.a.	111 ο	127

CG — Complex Greek

32	48	n.a.	80 Π	96	112 π
33	49	65 Α	81 Θ	97 α	113 ϑ
34	50	66 Β	82 Ρ	98 β	114 ρ
35	51	67 Η	83 Σ	99 η	115 σ
36	52	68 Δ	84 Τ	100 δ	116 τ
37	53	69 Ε	85 ϒ	101 ε	117 υ
38	54	70 Φ	86	102 φ	118
39	55	71 Γ	87 Ω	103 γ	119 ω
40	56	72 Χ	88 Ξ	104 χ	120 ξ
41	57	73 Ι	89 Ψ	105 ι	121 ψ
42	58	74	90 Ζ	106	122 ζ
43	59	75 Κ	91	107 κ	123
44	60	76 Λ	n.a.	108 λ	124
45	61	77 Μ	93	109 μ	125
46	62	78 Ν	n.a.	110 ν	126
47	63	79 Ο	n.a.	111 ο	127

IG — Indexical Complex Gr

32	48	n.a.	80 Π	96	112 π
33	49	65 Α	81 Θ	97 α	113 ϑ
34	50	66 Β	82 Ρ	98 β	114 ρ
35	51	67 Η	83 Σ	99 η	115 σ
36	52	68 Δ	84 Τ	100 δ	116 τ
37	53	69 Ε	85 ϒ	101 ε	117 υ
38	54	70 Φ	86	102 φ	118
39	55	71 Γ	87 Ω	103 γ	119 ω
40	56	72 Χ	88 Ξ	104 χ	120 ξ
41	57	73 Ι	89 Ψ	105 ι	121 ψ
42	58	74	90 Ζ	106	122 ζ
43	59	75 Κ	91	107 κ	123
44	60	76 Λ	n.a.	108 λ	124
45	61	77 Μ	93	109 μ	125
46	62	78 Ν	n.a.	110 ν	126
47	63	79 Ο	n.a.	111 ο	127

GG — Gothic German

32	48	n.a.	80 𝔓	96	112 𝔭
33	49	65 𝔄	81 𝔔	97 𝔞	113 𝔮
34	50	66 𝔅	82 ℜ	98 𝔟	114 𝔯
35	51	67 ℭ	83 𝔖	99 𝔠	115 ſ
36	52	68 𝔇	84 𝔗	100 𝔡	116 𝔱
37	53	69 𝔈	85 𝔘	101 𝔢	117 𝔲
38	54	70 𝔉	86 𝔙	102 𝔣	118 𝔳
39	55	71 𝔊	87 𝔚	103 𝔤	119 𝔴
40	56	72 ℌ	88 𝔛	104 𝔥	120 𝔵
41	57	73 ℑ	89 𝔜	105 𝔦	121 𝔶
42	58	74 𝔍	90 ℨ	106 𝔧	122 𝔷
43	59	75 𝔎	91	107 𝔨	123
44	60	76 𝔏	n.a.	108 𝔩	124
45	61	77 𝔐	93	109 𝔪	125
46	62	78 𝔑	n.a.	110 𝔫	126
47	63	79 𝔒	n.a.	111 𝔬	127

GI — Gothic Italian

32	48	n.a.	80 P	96	112 p
33	49	65 A	81 Q	97 a	113 q
34	50	66 B	82 R	98 b	114 r
35	51	67 C	83 S	99 c	115 s
36	52	68 D	84 T	100 d	116 t
37	53	69 E	85 U	101 e	117 u
38	54	70 F	86 V	102 f	118 v
39	55	71 G	87 W	103 g	119 w
40	56	72 H	88 X	104 h	120 x
41	57	73 I	89 Y	105 i	121 y
42	58	74 J	90 Z	106 j	122 z
43	59	75 K	91	107 k	123
44	60	76 L	n.a.	108 l	124
45	61	77 M	93	109 m	125
46	62	78 N	n.a.	110 n	126
47	63	79 O	n.a.	111 o	127

CC — Complex Cyrillic

32	48	n.a.	80 П	96	112 п
33	49	65 А	81 Ш	97 а	113 ш
34	50	66 Б	82 Р	98 б	114 р
35	51	67 Э	83 С	99 э	115 с
36 Ц	52	68 ц	84 Т	100 д	116 т
37 Ъ	53	69 ъ	85 Ю	101 й	117 ю
38 Ы	54	70 ы	86 В	102 ф	118 в
39	55	71 Г	87 Щ	103 г	119 щ
40	56	72 Ж	88 Х	104 ж	120 х
41	57	73 И	89 У	105 и	121 у
42 Е	58	74 е	90 З	106 ч	122 з
43	59	75 К	91	107 к	123
44	60	76 Л	n.a.	108 л	124
45	61 Я	77 М	93	109 м	125
46	62	78 Н	n.a.	110 н	126
47	63 Ь	79 О	n.a.	111 о	127

AR — Cartographic Roman

32	48 0	n.a.	80 P	96	112 P
33 !	49 1	65 A	81 Q	97 A	113 Q
34 "	50 2	66 B	82 R	98 B	114 R
35 °	51 3	67 C	83 S	99 C	115 S
36 $	52 4	68 D	84 T	100 D	116 T
37	53 5	69 E	85 U	101 E	117 U
38 &	54 6	70 F	86 V	102 F	118 V
39 '	55 7	71 G	87 W	103 G	119 W
40 (	56 8	72 H	88 X	104 H	120 X
41)	57 9	73 I	89 Y	105 I	121 Y
42 *	58 :	74 J	90 Z	106 J	122 Z
43 +	59 ;	75 K	91	107 K	123
44 ,	60 ‘	76 L	n.a.	108 L	124
45 —	61 =	77 M	93	109 M	125
46 .	62 ’	78 N	n.a.	110 N	126
47 /	63 ?	79 O	n.a.	111 O	127

AG — Cartographic Greek

32	48	n.a.	80 Π	96	112 Π
33	49	65 Α	81 Θ	97 Α	113 Θ
34	50	66 Β	82 Ρ	98 Β	114 Ρ
35	51	67 Η	83 Σ	99 Η	115 Σ
36	52	68 Δ	84 Τ	100 Δ	116 Τ
37	53	69 Ε	85 ϒ	101 Ε	117 ϒ
38	54	70 Φ	86	102 Φ	118
39	55	71 Γ	87 Ω	103 Γ	119 Ω
40	56	72 Χ	88 Ξ	104 Χ	120 Ξ
41	57	73 Ι	89 Ψ	105 Ι	121 Ψ
42	58	74	90 Ζ	106	122 Ζ
43	59	75 Κ	91	107 Κ	123
44	60	76 Λ	n.a.	108 Λ	124
45	61	77 Μ	93	109 Μ	125
46	62	78 Ν	n.a.	110 Ν	126
47	63	79 Ο	n.a.	111 Ο	127

C

Symbol Fonts

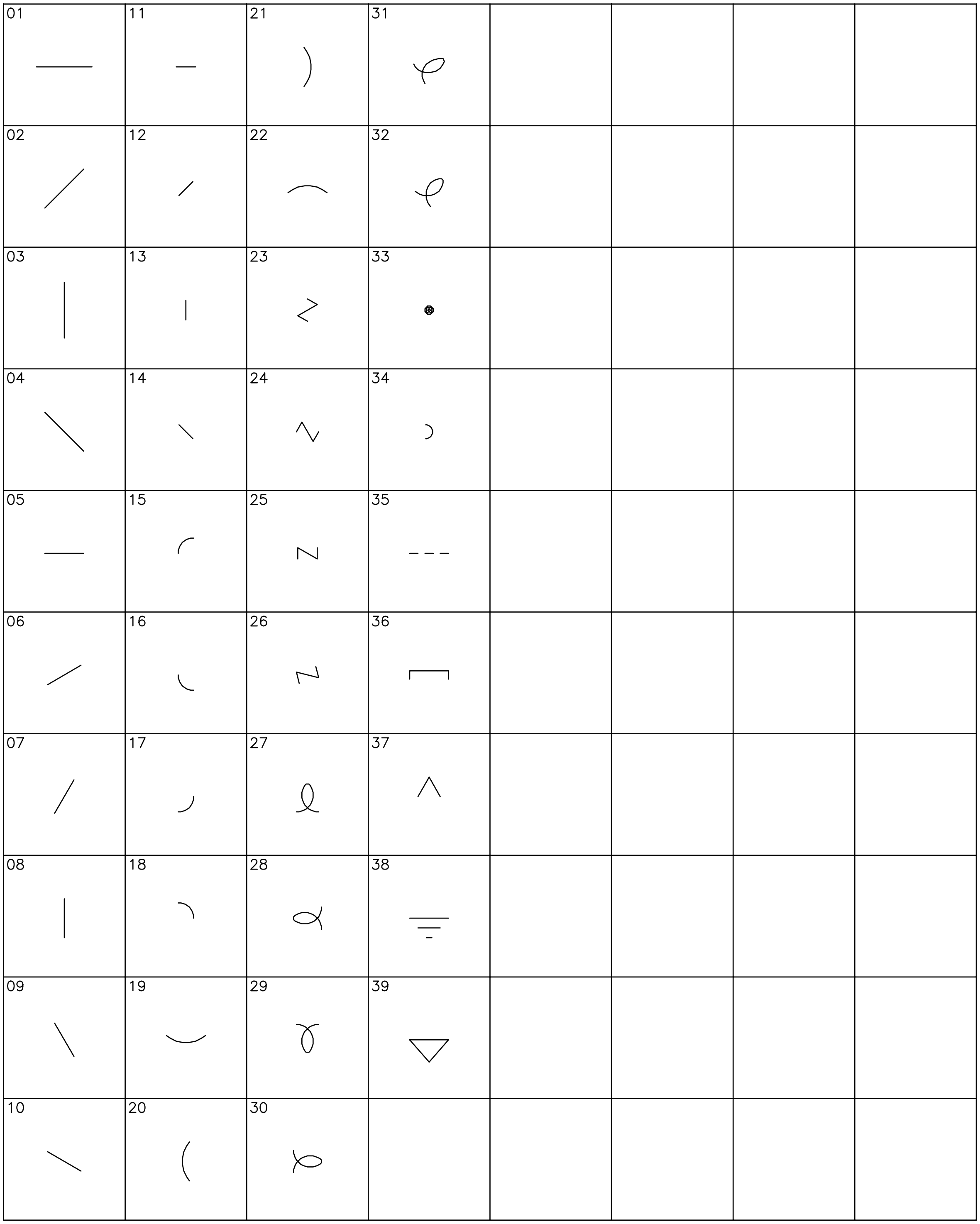
ZO – Zodiac

01 ☉	11 ☾	21 ♍					
02 ☿	12 ☄	22 ♎					
03 ♀	13 ✳	23 ♏					
04 ⊕	14 ☊	24 ♐					
05 ♂	15 ☋	25 ♑					
06 ♃	16 ♈	26 ♒					
07 ♄	17 ♉	27 ♓					
08 ⛢	18 ♊						
09 ♆	19 ♋						
10 ♇	20 ♌						

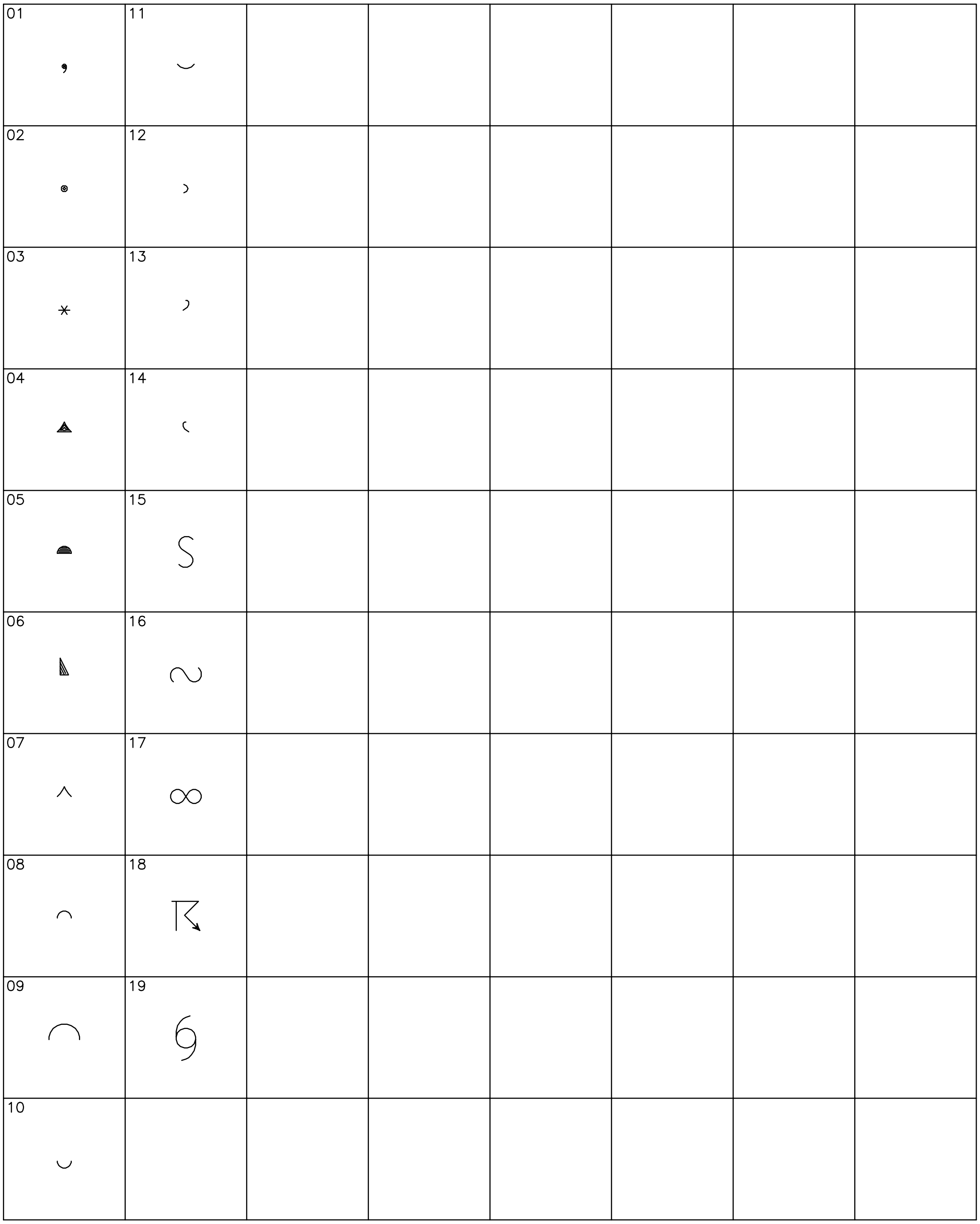
MU — Music

01 ·	11 𝄼	21 𝅗	31 𝄢				
02 𝅮	12 𝄽	22 ●	32 𝄡				
03 𝅮	13 𝄾	23 ♯					
04 𝅝	14 𝄞	24 ♮					
05 𝅗	15 𝄢	25 ♭					
06 ●	16 𝄥	26 𝄻					
07 ♯	17 ·	27 𝄼					
08 ♮	18 𝅮	28 𝄽					
09 ♭	19 𝅮	29 𝄾					
10 𝄻	20 𝅝	30 𝄞					

EL — Electrical



WE — Weather



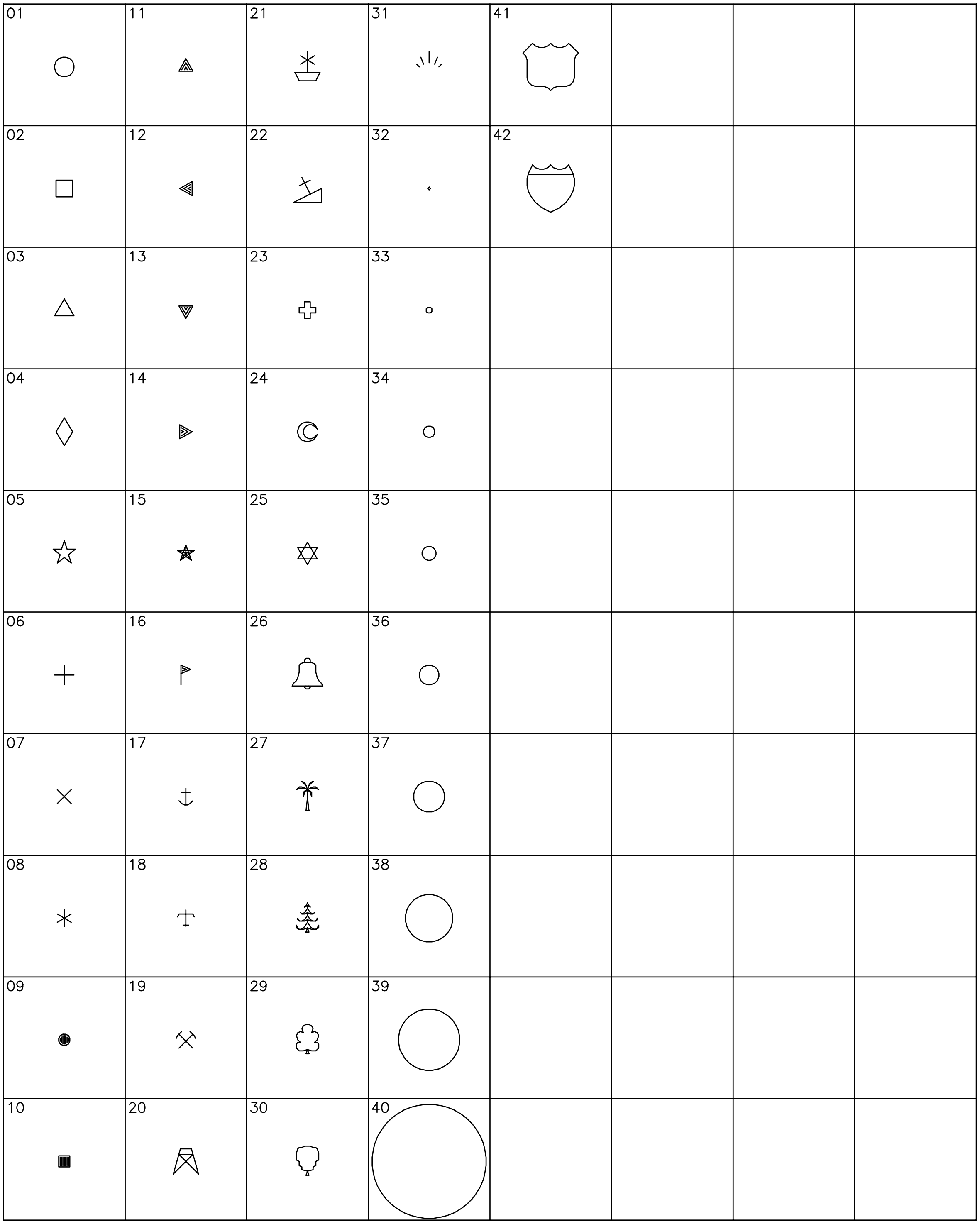
MA — Math

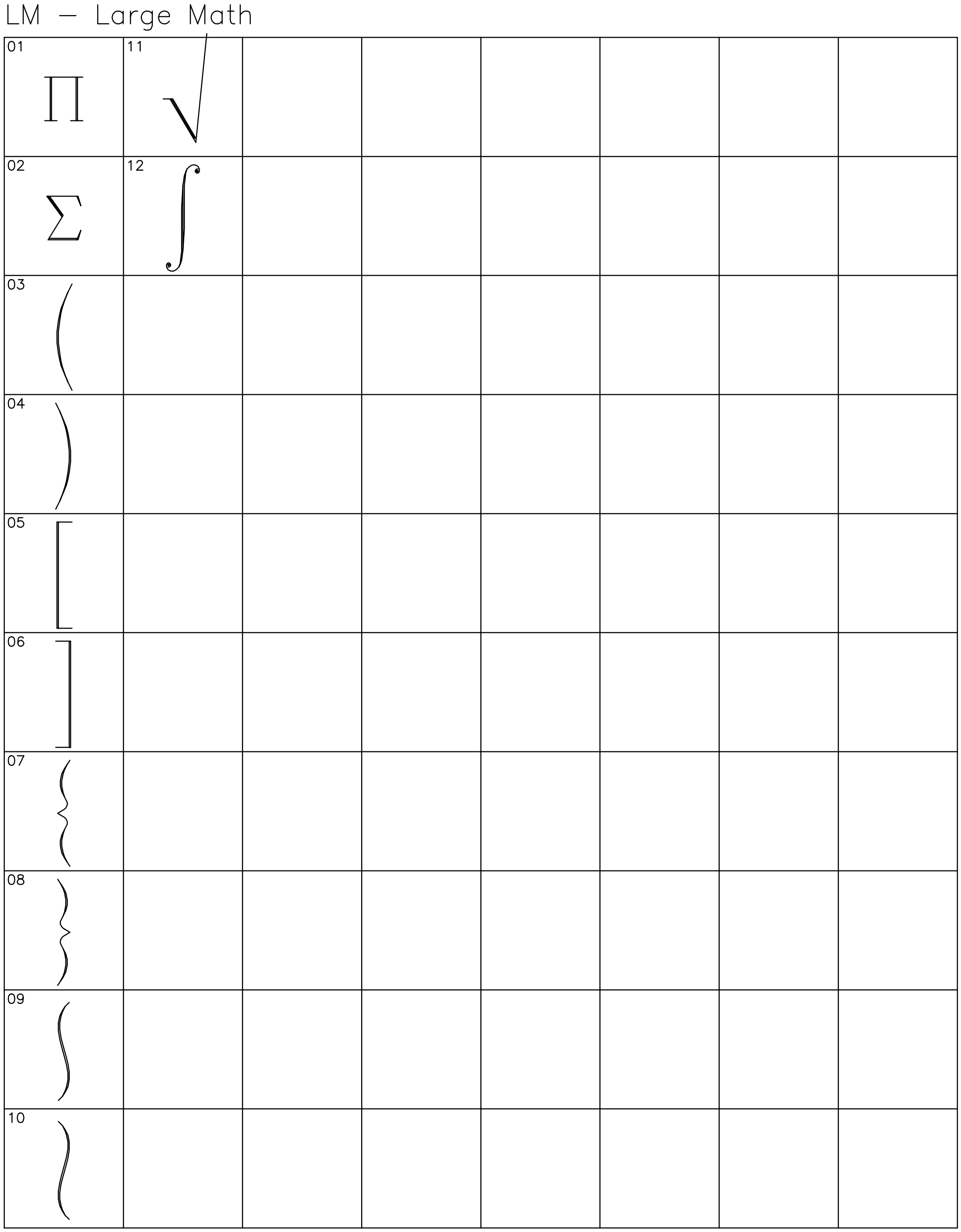
01 ℵ	11 ς	21)	31 +	41 >	51 ‘	61 ↑	71 &
02 ﬀ	12 ℘	22 [	32 ±	42 ≦	52 ‘	62 ←	72 @
03 ﬁ	13 *ﬀ*	23]	33 ∓	43 ≧	53 ’	63 ↓	73 $
04 ﬂ	14 *ﬁ*	24 {	34 ×	44 ∝	54 √	64 ∂	74 #
05 ﬃ	15 *ﬂ*	25 }	35 ·	45 ∼	55 ⊂	65 ∇	75 §
06 ﬄ	16 *ﬃ*	26 ⟨	36 ÷	46 ^	56 ∪	66 √	76 †
07 ı	17 *ﬄ*	27 ⟩	37 =	47 ´	57 ⊃	67 ∫	77 ‡
08 ϵ	18 *ı*	28 |	38 ≠	48 `	58 ∩	68 ∮	78 ∃
09 θ	19 /	29 ‖	39 ≡	49 ˘	59 ∈	69 ∞	
10 ϕ	20 (	30 —	40 <	50 ’	60 →	70 %	

SM – Simplex Math

01 ∇	11 ;	21 |	31 #	41 ♧			
02 υ	12 !	22 —	32 &	42 ☘			
03 ∂	13 ?	23 +	33 ⌑	43 ⚜			
04 ϵ	14 '	24 =	34 ‖				
05 θ	15 "	25 ×	35 ⊥				
06 ϕ	16 °	26 *	36 ∠				
07 ς	17 $	27 ·	37 ∴				
08 .	18 /	28 ‘	38 ♤				
09 ,	19 (	29 ’	39 ♡				
10 :	20)	30 →	40 ♢				

MP — Map


01
11
21
31
41
02
12
22
32
42
03
13
23
33
04
14
24
34
05
15
25
35
06
16
26
36
07
17
27
37
08
18
28
38
09
19
29
39
10
20
30
40


LM — Large Math
01
02
03
04
05
06
07
08
09
10
11
12

IZ — Indexical Zodiac

01 ☉	11 ☾						
02 ☿	12 ☄						
03 ♀	13 ✳						
04 ⊕	14 ☊						
05 ♂	15 ☋						
06 ♃							
07 ♄							
08 ⛢							
09 ♆							
10 ♇							

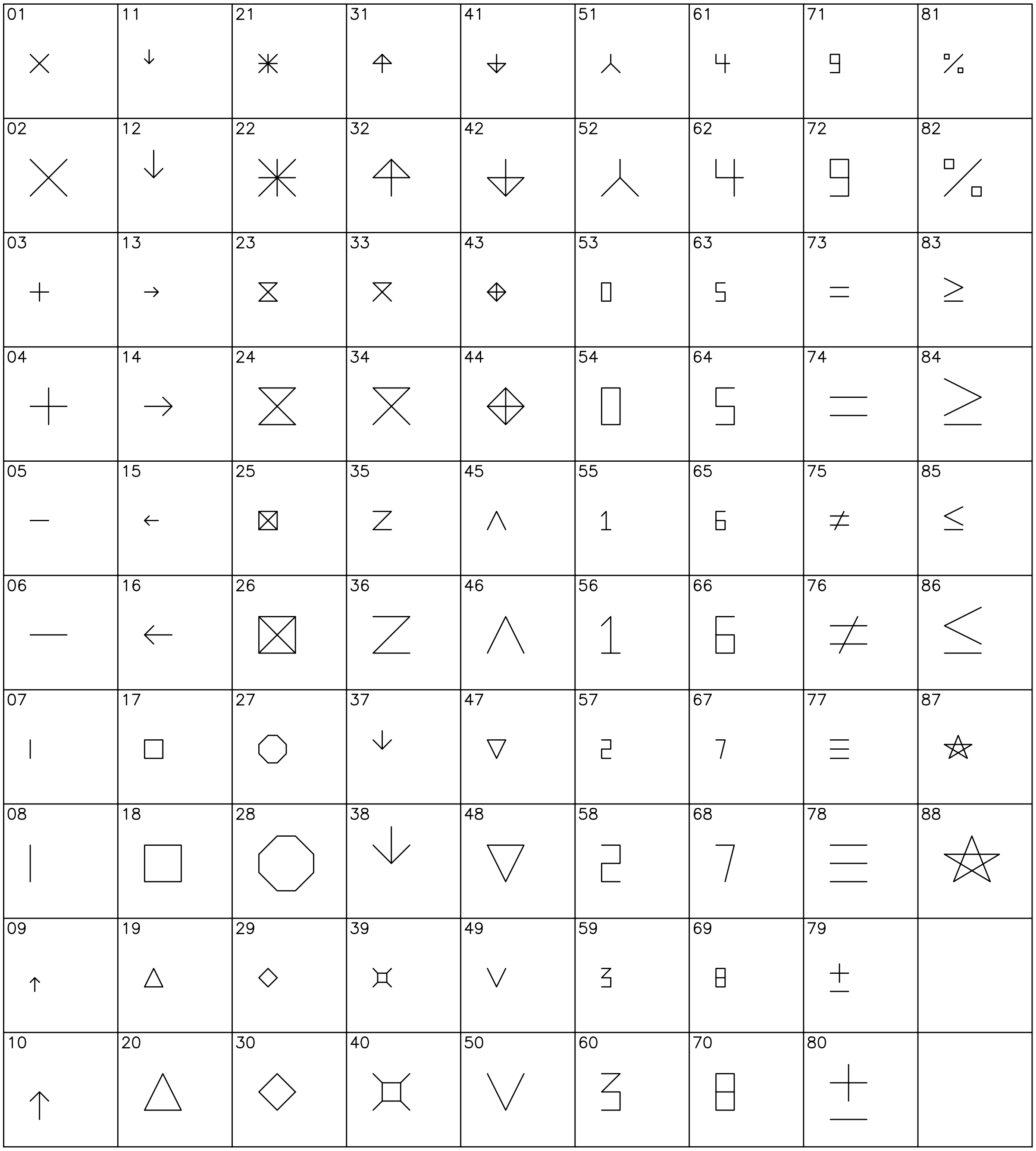
IM — Indexical Math

01 .	11 /	21 ‖	31 ≡	41 ˘	51 →	61 %	
02 ,	12 (	22 −	32 <	42 ’	52 ↑	62 &	
03 :	13)	23 +	33 >	43 ‘	53 ←	63 @	
04 ;	14 [	24 ±	34 ≦	44 ʿ	54 ↓	64 $	
05 !	15]	25 ∓	35 ≧	45 ʾ	55 ∂	65 #	
06 ?	16 {	26 ×	36 ∝	46 ⊂	56 ∇	66 §	
07 ′	17 }	27 ·	37 ∼	47 ∪	57 √	67 †	
08 ″	18 ⟨	28 ÷	38 ˆ	48 ⊃	58 ∫	68 ‡	
09 °	19 ⟩	29 =	39 ´	49 ∩	59 ∮	69 ∃	
10 *	20 |	30 ≠	40 `	50 ∈	60 ∞		

CA — Cartographic

01 .	11 /	21 ‘					
02 ,	12 (	22 ’					
03 :	13)	23 →					
04 ;	14 |	24 #					
05 !	15 —	25 &					
06 ?	16 +	26 □					
07 '	17 =						
08 "	18 ×						
09 o	19 *						
10 $	20 .						

PM — Plot Marks



Index